Playing videos in an augmented reality setting on an Android mobile platform

A thesis

submitted in partial fulfilment

of the requirements for the degree

of

Master of Science (Research)

at

The University of Waikato

by

Shuzu Li

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

2018

Abstract

Augmented Reality (AR) is typically used to superimpose pictures, texts or 3D models on a camera view of the surrounding reality. In this thesis, we use the video in an AR environment as a rich way to present information. This thesis explored the feasibility of playing elements of videos via Augmented Reality on an Android mobile phone platform. Our prototype, VideoAR, is introduced in this thesis. VideoAR presents the capability to manipulate videos in AR such as playing, shaping and background subtracting. It has been developed based on the Persona and Scenario method, and evaluated via observation and measurement methods. Overall, we were successful in overlaying elements of videos with camera views in an AR-like fashion. However, we identified a number of factors that can affect the quality of video playing in AR such as distance, angle, light, cover, texture, colour and shape. The observed results encourage the use of video in AR type environments on end-user mobile phones for industries like tourism, film, academia and architecture.

Acknowledgements

I pay my deep gratitude to Professor Annika Hinze, who was my supervisor, for her patient guidance, encouragement and helpful critiques for my research. Also, I want to thank my secondary supervisor, Professor Sally Jo Cunningham for her support and inspirations.

Special thanks should be given to my family for their encouragement and help throughout my study. Most importantly, I wish to thank my loving and supportive partner, Yuan Deng, who provides unending inspiration.

Table of Contents

List of Figures

List of Tables

1 Introduction

The context and the goal of this project are introduced in this chapter. This is followed by an overview of the thesis outline.

1.1 Context and Goal of the Project

Augmented Reality (AR) is a technology which can blend digital items with objects in the real world interactively and seamlessly (Azuma et al., 2001). AR has been used for tourist applications (Yovcheva, Buhalis, & Gatzidis, 2012), outdoor navigations (Dünser et al., 2012) and urban games (Broll et al., 2008). It offers an innovative way to provide interactive rich information. The video offers another way to provide rich information compared to 2D pictures or texts. Appiah (2006) observed that websites containing video gain more favourable responses from users than pure text- or picture-based websites. Given these advantages, there could be more possibilities to create a better user experience when combining technologies such as AR, multimedia, geo-location and mobile phones. This thesis explores providing video content related to a user's current location as the augmentation to what users see in the real world through their mobile phone's camera, in real time. The following examples briefly illustrate the idea and identify possible issues.

The tourism industry has a strong demand for the services based on integrated and dynamical technology for the purpose of expanding the market (García-Crespo et al., 2009). AR technology can meet this need by providing tourists with dynamic and new interactive experience when they are travelling (Kounavis, Kasimati, & Zamani, 2012). Many of categorised tourisms can take advantage of AR technology. For example, movie-induced tourism. Movie-induced film is defined by Macionis (2004) as a type of tourism that refers to a post-modern experience of a location linked to some forms of media representation. Singh and Best (2004) pointed out that forty percent of visitors are motivated to visit the Hobbiton Movie Set to experience the natural scenery of 'Hobbiton' after they watched the related films. In this case, they might be disappointed by the experience if they cannot recognize the movie set for some reason. For example, the natural landscape has changed over time or the set is covered by snow. Therefore, it would be desirable to introduce a tool that would assist tourists to identify these movie-sets and provide information related to the movies. By using AR, the user experience can also be lifted to another level. Ideally, users can identify the features in the Hobbiton Movie Set through the cameras of the mobile devices and see snippets of the Hobbiton Movie embedded in the correct position. The characters originally from the movie will possibly walk out and act in this real environment. This would let users experience the exact scene that the current view is linked to, which will fulfil their goals for visiting.

Hamilton Gardens can be taken as another example of tourism that could benefit from the use of AR technology on mobile phones. Hamilton Gardens has 40 years history, and its features have changed over time (Hamilton Gardens New Zealand, 2017). AR technology can recreate scenes from the past. For example, imagine that in one of the

gardens, you can see a woman in a 60s dress and a man wearing an old fur fedora talking or walking with her. All these imaginations can happen in the frame of the camera and possibly give a direct feeling about the historical changes.

AR technology for videos on mobile phones also would benefit the filming industry. It enables videos to be put into real soundings for prototyping purpose. For example, testing a movie scene by using real characters with the associated equipment and crew for the shooting can be costly and time-consuming. Simply shooting a short video with toy cars or dolls in it and placing the video into the real set could be an easy way to envisage the scene in the mind of the director.

The initial motivation of this research was offering video-based AR technology on mobile phones to visitors of popular tourist attractions like the Hamilton Gardens and the Hobbiton Movie Set in New Zealand. We think that combining video-based AR, mobile devices and geolocation technologies can significantly help tourists to link the visual information and the real objects. However, videos are not typically in AR, which often uses texts and images. This therefore defines the scope of this project.

The goal of this project is to investigate the feasibility of blending multimedia (e.g. a video) with real objects rather than images or a 3D model. For this purpose, a prototype software for displaying video in an Augmented Reality setting, called VideoAR, is developed.

1.2 Overview of the Thesis

In this thesis, Chapter 1 introduced the context and the goal to the thesis. Chapter 2 describes the Persona-Scenario methodology for identifying requirements. Chapter 3 outlines the requirements based on the personas and scenarios in Chapter 2. Chapter 4 compares and groups related systems and shows how these related systems solve the issues. In Chapter 5, the designs and considerations for the system are shown. In the following Chapter 6, implementations of key features are presented. Chapter 7 shows factors that can affect the system and limitations by an evaluation. In Chapter 8, we discuss the potential usages of our system in real life and how our system bridges the gaps compared to the related work. Ultimately, in Chapter 9, we summarise key points, draw the conclusion and disclose the future work to be explored in the future.

2 Persona and Scenario

In this chapter, two usability tools are employed for identifying the requirements for the system. The first is Persona. Personas are hypothetical users of the system, they are related to goals and the motivation of the personas want to use the system (Cooper, 2004). Once the significant users are known for the system, there is a need to know how they use the system and in what context. A scenario is then designed for this purpose. Scenario is defined as the description about how users interact with the system (Carroll, 2000).

By analysing personas and scenarios, the requirements can be abstracted subsequently. In this chapter, personas are assumed for the system design purpose, not based on qualitative or quantitative user research as they are aiming to discover the requirements of the system.

2.1 Persona

Since our initial motivation was to help tourists in tourist attractions like Hamilton Gardens and the Hobbiton movie set. The persona in this section was created to model a visitor to the above tourist attractions, who may be interested in using a mobile phone for interacting with the attractions. The aim is to demonstrate how this type of tourists use our system in the places.

With the use of a (potentially high-end) mobile phone and an interest in AR visualisations, we design our persona to be younger than the average tourist. Many visitors to attractions in New Zealand are from overseas (in addition to many visitors from New Zealand) and are inspired to visit the country because they have encountered its attractions in movies, such as the Lord of the Rings (Jackson, Osborne & Walsh (Producer), & Jackson (Director), 2001).

Audrey, 28 years old, single, lives with her family in England. She loves the movie The Lord of the Rings and anything magic-related. She likes to watch movies with her friends on weekends. She is working as an accountant in the CBD, London. She travels around the world on her annual leaves alone or with her friends. She loves hiking, taking selfies, and exploring locations of movies she likes. She uses mobile applications to share her selfies and personal life.

2.2 Scenarios

In this section, we use two scenarios to show how the system works for users. Note that while the scenarios are using certain design elements (such as markers on maps or bubbles, the radar, flags, etc), these are not meant to be prescriptive but rather illustrative means to convey the possibilities of the interactions. For example, instead of semi-transparent text bubbles, we could envision any other means of conveying textual or graphical information, for example, pop-up boxes, points, dots, etc.

.

2.2.1 Scenario 1: Visiting Matamata

A. Collect Information

Audrey's annual application finally is approved. She plans to go to New Zealand. She knows the movie The Lord of the Rings was filmed in New Zealand. She thinks it would be cool to go to the movie sets of The Lord of the Rings. She uses the system to look up the movie during the lunch time in a restaurant. The system tells her about the movie, the locations of movie sets and souvenir shops for the movie. She browses locations of the movie sets and marks the places she wants to go by using the system. She only has 15 vacation days. There are also many other places she wants to try besides the movie sets. She searches the places and plans for the trip online.

B. Head to the Movie Set.

Audrey flies to New Zealand and she wants to go to the Hobbiton movie set first. She uses the system to share the location of the movie set to Google Maps and navigates herself to the destination. She drives there; purchases the ticket and enters the movie set.

C. Recognize the Movie Set.

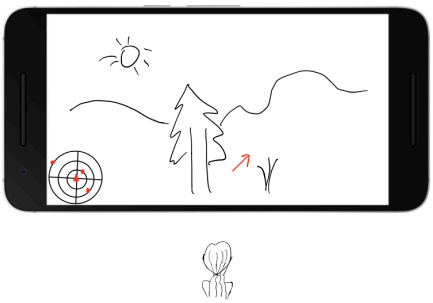


Figure 1: Find the near POIAR

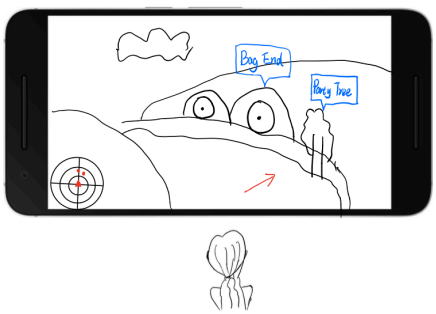


Figure 2: POIARs

Audrey walks around inside the movie set. She starts to use the system to find POIARs (Point of Interest in an AR view). Figure 1 shows a sketch of the view she sees when pointing her phone to her surroundings (shown in grey outline). She sees a red arrow on the camera pointing to the right side of herself. She sees a radar at the left bottom corner with three red blinking dots on it and a red triangle. She rotates herself until the triangle is pointing upwards of the phone. The positions of dots on the radar change accordingly when she is turning. She keeps walking forward while following the red arrow, then she sees that a text bubble is floating over the house and one over the big tree (Bag End is the house and Party Tree is the big tree) as Figure 2 shows. The arrow on the screen points at the tree. These text bubbles are semi-transparent. She notices the radar only has two dots left after she has walked here.

She wants to take a close look at the house, Bag End. She continues to walk towards it. As she is walking, the text bubbles remain the same size and floating over or on the house and tree all the time. She passes the tree, then the red arrow starts to point at the house. She keeps going towards the house. She sees the wooden door (see Figure 3). She finds the arrow disappears. The text bubble remains the same size and in the same relative position to the house.

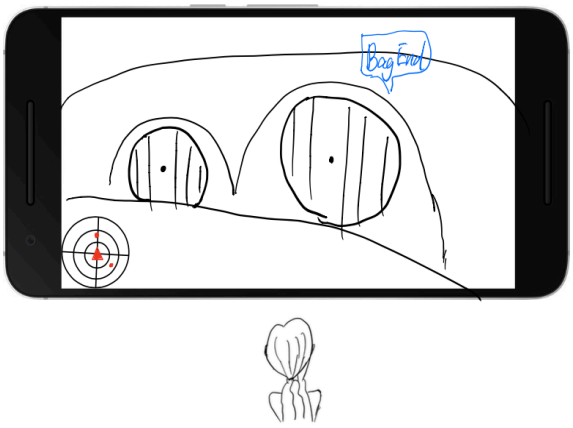


Figure 3: View an ARPOI closely

She wants to enter the house. There is a stepped path on the side to the house. She walks to the side of the house. She now only can see the part of the wooden door because a tree covers the door partly. The door looks not as round as it looks from the front side. However, she still sees the text bubble as Figure 4 shows in a blue colour.

Figure 4: View a POIAR from the side

D. Watch the Video

She is now in front of the door. She is holding the phone towards the wooden door. Then the door is opening and the hobbit, Bilbo Baggins (shown in green lines) from the movie comes out as Figure 5 indicates. When she moves her eyes out of her phone to the actual door (reality part in Figure 5), the door looks the same as before (as shown in blue lines).

She wants to take a close look at what is happening; she walks towards the door. Bilbo Baggins (shown in green outlines) still stands next to the door when she is walking as Figure 6 shows. However, the character and the door appear bigger on the screen compared to what she sees in Figure 5.

E. Capture the Magic

She wants to take a photo with Bilbo Baggins. She asks another tourist to take a photo for her. She shows the tourist how to do it. She stands in front of the door. The tourist takes a photo when the Bilbo Baggins comes out and stands next to Audrey. Audrey shares the picture to Facebook.

2.2.2 Scenario 2: Visiting Hamilton Gardens

After the tour described in Section 2.2.1, the sky is getting dark. She decides to stay in a nearby town. She drives into the Hamilton, checks in a hotel and orders her dinner. While waiting for the dinner, she thinks about what to do tomorrow. She will only have a half day because she has returned her rental car and needs to go to the next city by bus at 3pm next day.

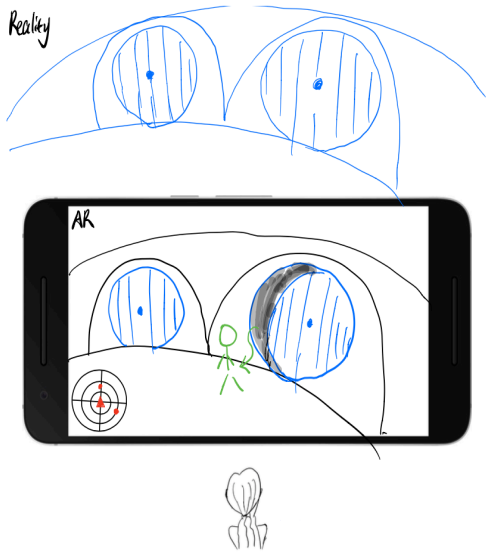


Figure 5: Watch the video in an AR view

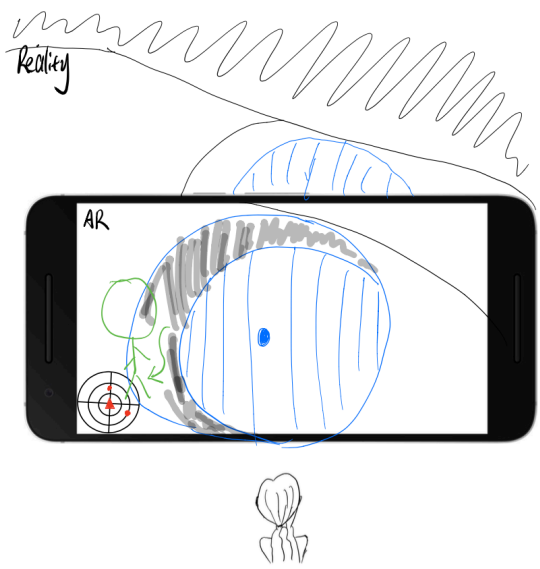


Figure 6: Watch the video closely in an AR view

A. Search nearby POIs

She uses the application to find if there are some interesting places around. All POIs are listed on a map in the system. She checks those places close to her. She finds an attraction named Hamilton Gardens within a walking distance from her hotel. She checks the pictures and the introduction of the Hamilton Gardens. She likes them and decides to take a tour there in the morning of the next day.

Figure 7: Map of POIs (The map part is adapted from Google Maps)

B. Go to the Hamilton Gardens

Audrey shares the location of the Hamilton Gardens from the system to Google Maps and walks to the target. Once she is there, she starts to navigate herself through the gardens. Firstly, she opens the map of the system as can be seen in Figure 7. She sees many red flags with texts. She finds that the location of the red triangle is her current location. She clicks the Chinese Scholars Garden flag. The flag then turns into a balloon. She now starts to use the AR view to navigate herself as Figure 8 represents.

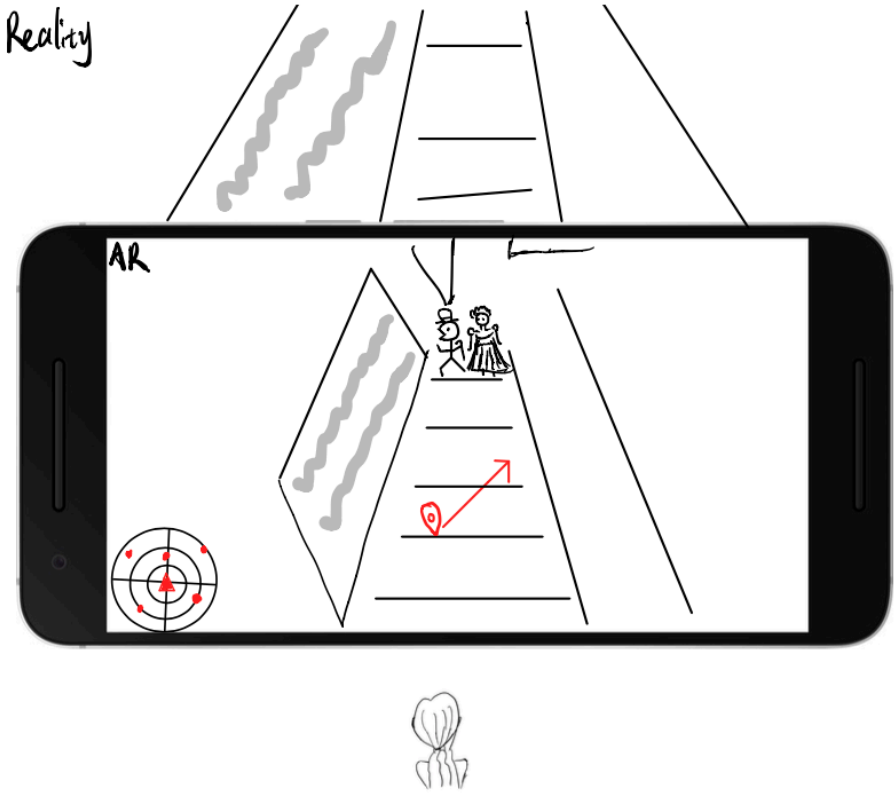


Figure 8: Seeing people dressing in old styles in Hamilton Gardens

C. Walk in the Hamilton Gardens

As shown in Figure 8, Audrey looks at the phone and starts to navigate in the AR view. She sees a red arrow pointing at the right side. There is a balloon at the end of the red arrow. She has seen the same balloon shape when she clicks the Chinese Scholars Garden flag in Figure 7. She sees two people coming. The man is wearing a fedora and holding a walking cane. He is talking with a lady in an old-fashioned dress. Audrey thinks the dress looks similar with her grandmother's. She finds the couple only exist in the camera view. She passes them and keeps following the red arrow.

D. Destination

She looks around while walking. She sees other gardens on the way to the Chinese Scholars Garden. She finds that there are flags in front of other gardens' gates as Figure 9 shows. She keeps walking and now can see the gate in Chinese style not far away. She sees that there is a balloon in front of this gate. She walks closely, then she finds that it is the Chinese Scholars Garden.

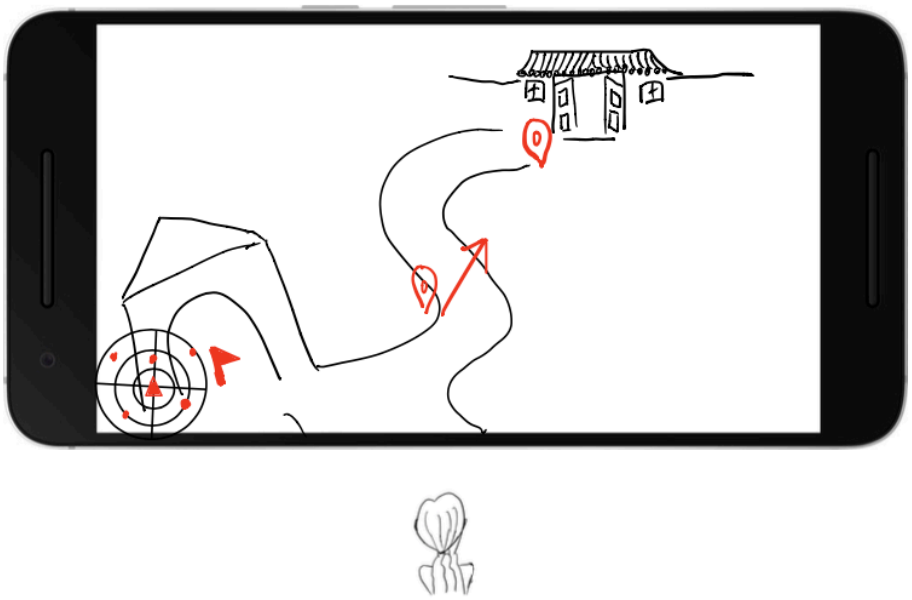


Figure 9: The flag and the balloon

E. More Tour Information about the Garden

She wants to know more information about this garden, so she interacts with the flag on the screen of her mobile phone. The system shows relative texts, images and videos. She learns more about the garden. She records what happened in the Hamilton gardens and wants to show the video to her family when she is back.

2.3 Summary

By presenting two scenarios and one persona, we show the idea of how our system works. Then the requirements abstracted according to the scenarios will be listed in following Chapter 3.

3 Requirements Analysis

Based on the scenarios for Audrey in the last chapter, relative the requirements are analysed and identified. Several requirements refer to the persona and scenarios (user requirements) in Chapter 2, while others are strongly influenced by the technical setting of AR (technical requirements) on mobile phones. In the summary of this section, we discuss which of these features will be implemented as part of this thesis, because our goal is to prove the feasibility of playing videos in AR in this research and not the development of a complete product. A naming system is designed for referring to the persona and scenarios sections in Chapter 2, for example, Step 1A means Step A in Scenario 1. R1 means Requirement 1 in this section.

3.1 Augmented Reality and Video

This section is the focus of the thesis. It presents how the system makes use of AR technology and what the system is capable to do. The section is divided into five parts. They are related to the AR technology that users are using in terms of navigation in an AR view, identification of AR targets, presentation of information and video blending and scaling.

3.1.1 Navigate in an AR View (R1)

To find POIARs (Point of Interest in an AR view), Audrey needs a navigation system. The system should help her find either nearby POIARs or a specific POIAR. In Step 1C, the system shows a navigation arrow to indicate the location of POIAR. There is only one arrow showing in the AR view at one time and it points at the closest POIAR. Once Audrey passed the POIAR for a certain distance that was targeted, the arrow will point at the next closest POIAR. If Audrey stands in a range that the system thinks it close enough to find the target by herself, the navigation arrow will disappear. The arrow will reappear when Audrey leaves out of the range.

Compared to the arrow navigation, the radar navigation provides more information about locations of the nearby POIARs than the arrow. The radar at the lower left corner in an AR view shows the location of POIARs around Audrey in a certain range, for example, 100 meters (shown in Figure 2, Step 1C). Andrey, therefore, knows what other POIARs are available and prepare her next stop. If Audrey uses the arrow navigation and radar together, locating the nearest POIAR can take less efforts than using bare eyes to look around as Audrey knows her surroundings and where to go.

For navigation to a specific target in a bigger range, a map in Figure 7 in Step 2A can be useful. In the case of Step 2A, Audrey is about to enter the Hamilton Gardens. Audrey needs to know what POIARs are available in order to plan how to go there. The map shows all POIARs with red flag markers. If Audrey sets one POIAR as the destination, the flag on the POIAR of the destination will be transformed into a balloon. The same balloon will be shown at the end of the arrow in an AR view for indicating the directions of the destination. Once the system detects the user in a close range to the destination,

the balloon will turn back into a flag (Step 2D). The radar and arrow are designed to be semi-transparent for a better user experience and safety reason. As users can still see the objects clearly through the radar and the arrow, they can avoid crashing into others. This requirement is abstracted from the persona and scenarios as outlined above (Step 2D and Step 2A) and considered here as a *user requirement*.

3.1.2 Identify the POIARs (R2)

The system should identify POIARs despite the factors such as distance, angle and cover of the surface. It is important that the system can recognize POIARs from a distance. In Step 1C, Audrey finds the Bag End and the Party Tree from a far distance as Figure 2 shows. In the same step, Figure 3 describes that Audrey walks closely to look at the wooden door. Regardless of distance, text bubbles are floating over the POIARs in the AR view.

Angle is another factor that the system should consider. In Step 1C, Audrey walks to the side of the house. The Bag End text bubble is still floating over the house, even from a side angle. In the same Figure 4, the tree covers a part of the house. The system should be able to identify the POIAR in this case.

To recognize the scene in the cases above, geolocation and photo recognition technology can be useful. Geolocation like users' location can be collected by the sensors embedded on mobile devices such as a GPS sensor, a compass sensor, an in-depth sensor. By knowing the user's current location and where the user is facing, it is possible to calculate what POIARs around the user and which direction the user is facing, even if POIARs are covered or out of eyesight of the user. Photo recognition will help with accurate recognition for the real objects; this will be the main way to identify the POIARs of our system. Geolocation technology will assist the photo recognition when the photo recognition fails or needs to be more accurate. However, the requirement is a *user requirement* as it is developed based on user interactions in the scenario 1, Step 1C.

3.1.3 Label the POIARs (R3)

The system has three types of markers to label POIARs in an AR view. They are text bubbles, flags and balloons. In Step 1C, Audrey sees the text bubbles over the tree and the house. The size of text bubbles and relative positions to the POIARs do not change when Audrey is approaching to the house and the tree. The system uses text bubbles to label nearby POIARs. In Step 2D, Audrey sees a flag standing in front of the garden gate and a balloon shown next to the destination garden, Chinese Scholars Garden. Both remain their relative positions to the POIARs. However, their scale changes according to the size of POIARs identified in Step 1C. The system uses flags for nearby POIARs and a balloon for targeted POIAR when the user has set a destination by Map feature (R7) in Section 3.2.2. The system also labels POIARs on the radar (R6) which will be discussed in Section 3.2.1. The requirement is identified as a *user requirement* which is linked to the user interactions in Step 2D and Step 1C.

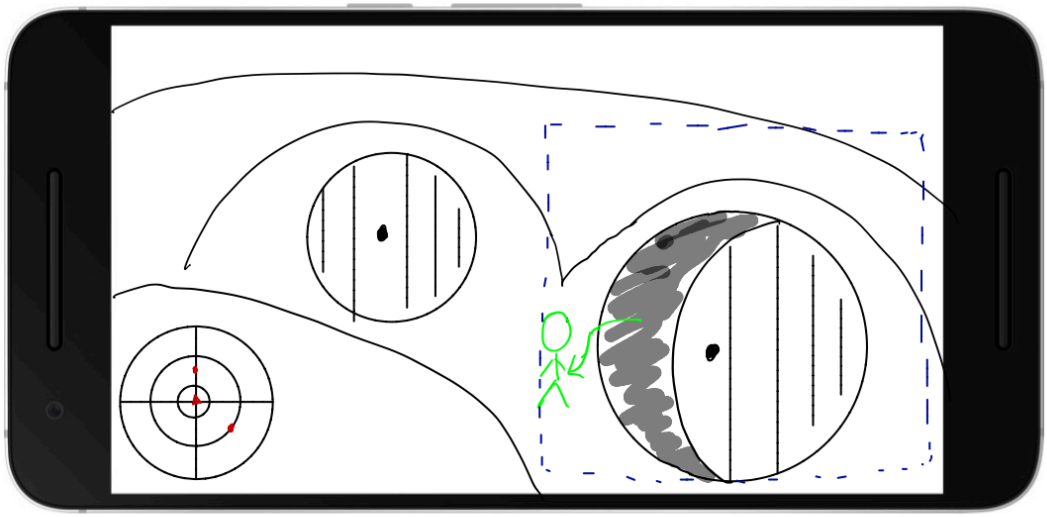


Figure 10: Mode I - Video overlay

3.1.4 Blending Videos into Reality (R4)

Audrey sees that Bilbo Baggins opens the door and walks out in Step 1D. Audrey also sees a couple in old fashioned clothes walking and talking in Step 2C. These scenes only happened in the AR view on her phone. To achieve these, the system should have the ability to play a video in an AR view. However, there are several modes to superimpose a video on real objects based on the video types. The system aims to blend a video into reality. Three modes following are designed for this purpose. Each of them tackles different difficulties and each extends previous mode's capabilities.

The first mode is designed to place a video in an AR view in the correct position and scale. From the Step 1D, Audrey sees the character coming out from the round wooden door as Figure 5 and Figure 6 describe. What Andrey saw is a video clip from The Lord of the Rings. In the video clip, Bilbo Baggins opens the door. In reality, the system identifies the same round door (Identify the POIARs, R2), then adjusts the scale of the video for overlapping the door in the video with the real door as dotted area (as Figure 10 shows). The dotted area is square because the background and boundaries of the video is square. In this mode, the system does not manipulate the background and boundaries of the video clip.

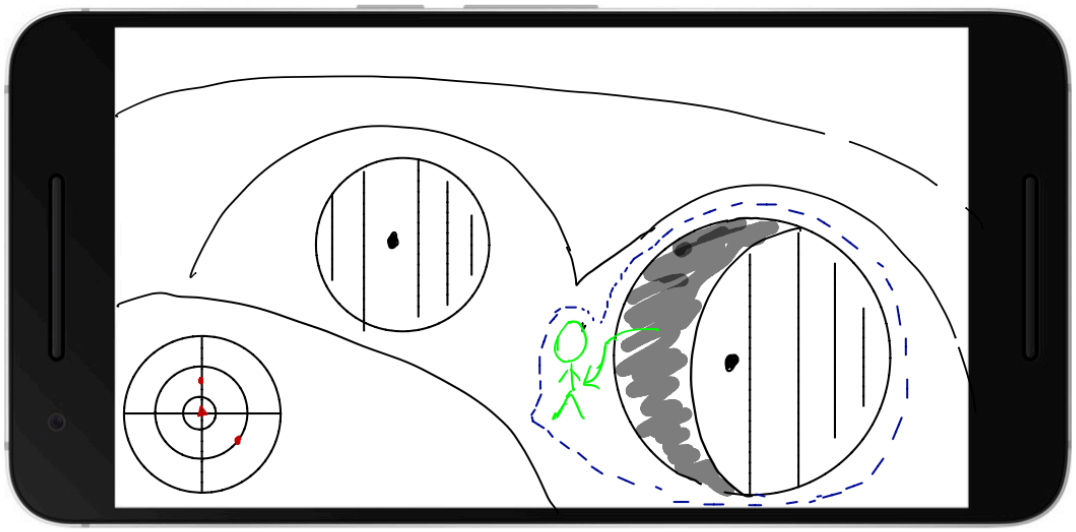


Figure 11: Mode II - Video reshape

To extend mode I, the second mode tries to show the real objects as much as possible in an AR view. As Figure 11 shows, the video clip covers the real round door in an AR view. Nevertheless, the shape of the dotted area is irregular as the video has been reshaped by the system. The system removes a part of the background from the video clip in order to expose the real surroundings to the maximum degree. In this mode, the video appears to be more naturally in reality from the camera view. The video clip used is the same with the one used in Mode I.

The last mode is referring to the Step 2C. In which, Audrey sees a couple walking and talking. The walking man and the lady are originally from a video. In this mode, the system focuses on removing the background of the video completely. As shown in Figure 12, the man and woman are circled by dashed lines. That is the only part of the video showing in the AR view. However, as the couple are moving, the video still has boundaries and a transparent background. However, those are invisible to users. The requirement 4 is a *technical requirement* as the AR technology is required by our technical settings in Step 1D and Step 2C.

3.1.5 Dynamic Ratio (R5)

From the Step 1D, Audrey walks towards the wooden door while watching the video in an AR view as shown in Figure 6. Compared to what Audrey saw in Figure 5, the character and the door are bigger in the AR View. This is because the ratio of the video has been changed to fit better in the real environment. For example, when moving a camera close to the real objects, the objects should generally look bigger in camera. This works in the same way to the zoom feature. As VideoAR is placing a video in the AR view, the ratio of videos to real objects should be kept constant otherwise, users may see the video over-covering the real objects or being too small to watch. This feature can potentially offer better user experience in terms of blending the virtual movie with the

movie set. This requirement improves R4 in terms of user experience in Step 1D. It is considered as a *technical requirement*.

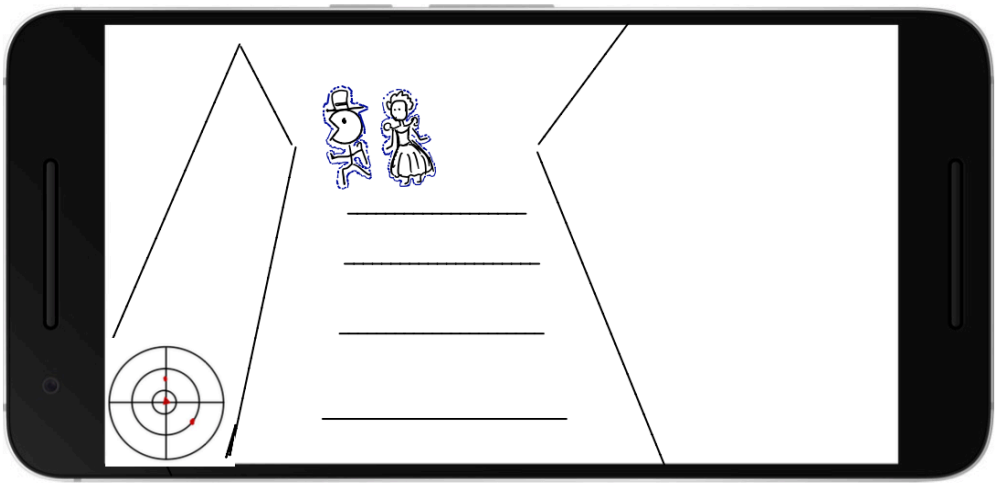


Figure 12: Mode III - Video background removal

3.2 Geolocation

This section explains the requirements related to geolocation technology. Some features support the application in terms of geo-data. Others allow users to be context-aware.

3.2.1 Radar (R6)

Radar is used for indicating nearby POIARs when the user is using the AR view to navigate. It appears at the left bottom corner of the AR view in Step 1C, 1D, 2C and 2D. It uses the data offered by GPS sensors to calculate the directions and locations of nearby POIARs. The point of the triangle is always pointing at where the user is facing. The blinking dots signify the POIARs. The radar is updated regularly when the user is moving. Clearly, we can see that it is a *user requirement* because it helps users to navigate themselves in Step 1C, 1D, 2C and 2D.

3.2.2 Map (R7)

From the Step 2B, Audrey decides to go to Hamilton Gardens. Audrey needs to know what POIARs are available from a bird view in a bigger range. The map shown in Figure 7 is a sample of how the system works for this. The flags on the maps indicate POIARs (Point of Interest in an AR view); the triangle illustrates the current location of the user. Geo-location data is collected by GPS sensors of the mobile devices. Once the user selected a POIAR as the destination. The flag of the POIAR changes into a balloon. The map is also can be used for Search and Browse POIs (R9) with different markers. The requirement can be a *user requirement* as it is needed from our persona and scenarios related to Step 2B.

3.3 Mobile

In the section, the requirements about mobility are discussed. Since the system is designed for the public, the device should be carefully considered.

3.3.1 Handheld Device (R8)

Audrey needs a device that enables her to gain information without limitations of physical locations. This can refer to the scenario Step 1A, in which Audrey looks up the movie during her lunchtime. The device should also be lightweight. From Step 2B, Audrey cannot take too much weight because she is walking. The device ought to have multi geo sensors for supporting the system requirements (R2, R6 and R7). Audrey is travelling as a tourist. The device should not have a too particular outlook to draw too much attentions so as to put pressure on her. Her purpose of travelling is supposed to be pressure-released after long time working. While this requirement may be seen as a user requirement, we consider it in this thesis a *technical requirement* because we expect AR technology to run on a mobile device as described in Step 1A and Step 2B.

3.4 Information Management

What information and how users can manipulate the data is presented by requirements in this section. Users can actively interact with the system. The system also is capable to communicate with other systems.

3.4.1 Search and Browse POIs (R9)

From Step 1A, Audrey needs to have the ability to search and browse the POIs easily. Audrey is doing this during her lunchtime, so Audrey probably only has one hand available as the other hand is holding food. The system should offer the bookmarking features for marking and saving the POIs Audrey prefers. In case Audrey wants to know more (Step 2A, Step2E), the rich information should be presented in various ways such as text, images and videos. A map feature like the one in Map (R7) should be presented in the system. The markers indicating POIs in R9 are distinctive from the ones in Map (R7). The map should let users to search and locate POIs, even if they are not close to the POIs. The requirement 9 is a *user requirement* as it relates to the need of users for information in Step 2A and Step 2E.

3.4.2 Share to Google Maps (R10)

Audrey needs to have a means to navigate herself to the destination (Step 1B and Step 2B). The system should support sharing a POI location to Google Maps. This enables users to have possibilities to navigate either by driving or walking when they are far from the POIs. The requirement is seen as a *user requirement* since it helps users to navigate themselves in Step 1B and Step 2B.

3.4.3 Share to Social Media Apps (R11)

Audrey shares the photo over the Facebook (Step 1E). This shows that the system needs to communicate with social media apps, for example, Facebook. A feature like this can potentially entertain the users as they may like to share the amazing moments with her friends or family during the trip. It is a need from users, so it fits into the *user requirement* category (Step 1E).

3.4.4 Capture Photos (R12)

Step 1E illustrates that the system should enable users to capture a photo of the camera view anytime. This is because Audrey desires to take a photo that shows herself and the virtual character. Besides that, the feature should be easy to use and understand quickly as Audrey may ask someone other for help. The person does not want to spend much time on understanding the feature at the scene. This can be deemed as a *user requirement* based on the user need in Step 1E.

3.4.5 Record Videos (R13)

Audrey wants to share the trip experience with family or friends (Step 2E). Audrey wants to capture the moment not only by static photos but also videos, as videos are more vivid. Audrey needs a system that can record anytime, anywhere. The recorded videos should be saved permanently. It is a requirement related to the need of the user in Step 2E, so it should be a *user requirement*.

3.5 Summary

The requirements shown in this chapter present an overview of the use of the system and how the system should behave in the scenarios (Chapter 2). As Table 1 shows, we group requirements into two categories. R4, R5 and R8 are technical requirements because they are required by our technical setting of AR. Other requirements are abstracted from our persona and scenarios in Section 2 as user requirements. Based on these requirements, we review the related systems in following Chapter 4 and analyse if they support these requirements.

	Category	
Requirement Name	**User Requirement**	**Technical Requirement**
Navigate in AR View (**R1**)	●	
Identify the POIARs (**R2**)	●	
Label the POIARs (**R3**)	●	
Blending Videos into Reality (**R4**)		●
Dynamic Ratio (**R5**)		●
Radar (**R6**)	●	
Map (**R7**)	●	
Handheld Device (**R8**)		●
Search and Browse POIs (**R9**)	●	
Share to Google Map (**R10**)	●	
Share to Social Media (**R11**)	●	
Capture Photos (**R12**)	●	
Record Videos (**R13**)	●	

Table 1: Requirement categories. ● means the requirement fits into the category

4 Related Work

In this chapter, several systems related to AR technology, mobile or/and geolocation are explored from a technology perspective and regarding their support for the requirements identified in Chapter 3. The systems are grouped into two categories according to their ability to play videos in the AR view. Then, they are compared in four aspects, AR and Video, Geolocation, Mobile and Information Management in the summary section. By doing so, VideoAR can better investigate the issues discovered on an escalating level and cover the gap between these related systems.

4.1 Superimposing Videos on an AR View

In this section, systems that are able to overlay the reality with a video are explored. Users can see the reality and virtual videos playing simultaneously. These systems will be explained in terms of principles, designs and related technologies. Furthermore, a comparison between each of them and VideoAR will be also shown from a technical perspective. All systems in this section are shown in Table 2 as well.

Arakawa, Kasada, Narumi, Tanikawa, & Hirose (2013) presented another way to watch a video via AR technology. For the ease of referring to this system in this thesis, a name is given to it, **Camera Operator AR**. The system runs on a device. It is unclear if it is running on a mobile device (R8). However, judging from the figures in the work, the device is likely to be a mobile tablet. It allows users to view the movie set from the camera operator angle. Users can move and rotate themselves in the same way as the camera did in the movie (R1). The system also assists users to move and rotate as regards directions, angles and range by the formula and the design. The system first recognises the movie scene by comparing the first frame of the video with the current content of the camera when the device is pointing to the real set (R2). Once the scene is identified, the real objects are overlaid with the video. The video is played when users are moving or rotating. Moreover, the system plays video frame by frame accordingly. As videos could be filmed from different angles, the users have to rotate themselves to view the video from the same angles with frames' in the video. With regard to how to calculate angles for the videos and users, a gyroscope sensor is used to detect the angle rotated by users. The angles of cameras in the films are manually calculated. A formula is proposed for guiding users to rotate as expected. The formula can detect if the device is out of range of the expected angles. The frame of the video freezes when the device is out of range of the expected angles. The screen of the device may stop at half real objects and the half video. By this design, the user may realise that they need to rotate back to carry on. When the operator camera in the videos is moving, the user should move forwards or backwards accordingly. In doing this, the frames of video move in corresponding directions. However, users' behaviours are as expected as seen from the evaluation of the system in a railway museum.

The Camera Operator AR fails to label the movie scene (POIARs) in the requirement (R3) as it has no markers to label a scene. It plays videos frame by frame and overlaps the frame with the real scene only when the user is standing at a specific watching point

It cannot keep the correct ratio of videos to the real environment since the frames of the video occupy the whole screen of the device (R5). Hence, it does not really blend video into reality (R4), it overwrites reality. On the other hand, the system does not support any geolocation features (R6 and R7). Since users passively interact with the system, they have no control over the information, thus it fails to meet the requirements from R9 to R13 (Information Management, Section 3.4).

The Augmented Video Wall (Baldauf & Fröhlich, 2013) is presented on an Android system to allow users to interact synchronously with a public display in an AR view (R8). It has three modes for the collaborative experience. When a user is pointing a mobile phone at a monitor with preview pictures, the application running on the phone shows the monitor in its camera view. The user clicks one of the pictures on their phones. The monitor in the camera view starts to play videos according to the selected picture along with different modes, then superimposes the corresponding video on the public monitor.

In the Competitive Mode, users share the same control of the video playing on the monitor. The android phone acts like a remote control. The video that is being watched by a user can be paused or played by another user. The video on the monitor is visible to everyone. The second mode is Concurrent Mode. In this mode, users have an isolated video watching experience. The public monitor behaves differently for everyone. Users can control their own videos only and are unaware of others' actions. The difference between Concurrent Mode and the third mode, Socially Concurrent Mode, is the social feature. Users still have full control of their own videos on a public monitor, but they can see which video is watching by others currently and which one is popular. This social information is shown on the public sharing monitor. The videos that are being watched by someone are bordered with the red colour. The Likes number on the preview pictures indicates the popularity of the videos. The system is composed of two parts. The Android application on the user's phone operates like a remote control which is using Vuforia toolkit[1] for AR features. The software on the public monitor responds to the requests from the Android application by the TCP sockets protocol. The video data are stored on the mobile devices for smoothly playing. As to the evaluation, most users disliked the Competitive Mode and sometimes they were even irritated due to the interruption from other users. The Concurrent Mode was preferred by the majority according to this work. The issue of the system was privacy concern from users. The Socially Concurrent Mode reveals that people were unwilling to let others know what they were watching. Furthermore, the participants for evaluation were all from a tech event and most of them had technology background. Overall, the application proves the AR visualizations and serves the concurrent interactions tasks well.

In comparison to our requirements, this system lacks context-aware ability (R6 and R7), although it can play videos in the AR view. It only can recognise the pictures that are pre-set and be incapable of outdoor scenes like landscapes (R2). Similarly, it cannot

[1] Vuforia toolkit is a library for AR technology, https://www.vuforia.com

label POIs as it has no need to label those pre-set pictures (R3). The ratio is kept correct when playing videos, despite it only laying videos on the monitor area (R4 and R5). Moreover, it provides no navigation ability to users when they are using AR-based features (R1) despite taking advantage of Android mobile and network technology. Users need to stay still when using the application. The system cannot share data to other applications in terms of geolocation or photos (R10 and R11). However, users do not need to move in a large range when using the application as there are no POIs information to browse (R9). It neither can capture photos nor record videos as shown in Capture Photos (R12) and Record Videos (R13).

Film-location-AR developed by Park and Woo (2015) provides a novel way to watch film-clips via a geo-location and AR based navigation service. The name Film-location-AR is named for referencing in this thesis as the system does not have an official name in the work. The 5W1H metadata schema is applied to the management of data. The application is installed on a mobile device (R8) with extra embedded sensors. When users launch the application, it passes the user information and locations to the server by network. After processing these data, a server returns the tour information including videos, routes, POIs, stories.

There are four main features of the application. The first one is Story Navigation which shows a map (R7) of POIs and storylines. Users can either follow the routes of storylines in an AR View (R1) or go to the POIs nearby (R9). The server designs the storylines for users according to users' reviews. Therefore, the storylines are not fixed. A route recommended time is also shown to users for the tour assistance. The second feature, Watching Point Navigation and Playing a video Clip, helps user watch video snippets in a way that how the director filmed the video. The application guides users to a watching location and shows the identified movie scene (R3), then the user can watch the video clip superimposed over the counterpart in reality. Users can adjust the scale and the transparency of the video layer. The third feature (Acting Point Navigation and Pose Guide) is about photo capturing and video recording for mimicking the actor's gestures in the movie (R12 and R13). A bordered shape of the actor's gesture is laid over the movie set (R2). Users can put themselves into the shape to take a picture of acting like what the character did in the movie. The last feature (Augmented Content for Reproducing a Scene) reproduces the movie scene. The movie set can be different from the original scene due to the weather or other factors. This feature makes the current movie set more like the original. For example, the movie was filmed in winter and the ground was covered with snow. Users, however, still can see snow in the real movie set in summer. The application covers the ground with 2D snow in the camera view. All the navigation can be viewed from the AR view. Users can share the content generated by the application with each other. However, the system emphasises on that 5W1H metadata eases the management of data. The challenge to the system is how to improve the accuracy of the context awareness. More external sensors would be explored in the future. The system will also be evaluated in qualitative and qualitative experiments.

Film-location-AR does not fully support the requirements, Blending Videos into Reality (R4) and Dynamic Ratio (R5). It can overlap a video with a corresponding object in reality at the appropriate scale and position, which is quite similar to the Mode I (Original Overlay) in R4. The difference with the Mode I is that the user needs to walk to an exact watching point to watch the video. If the user walks closer or further, the video will fail to keep the right ratio (R5). The other two modes (Mode II-Video Reshape and Mode III-Video Background Removal) from R4 are not covered by this system. The system has a map and uses a GPS sensor to locate the POIs. It does not have a feature akin to Radar (R6) to inform users of nearby POIARs when navigating in an AR view. The Film-location-AR cannot talk to other applications like Google Maps (R10) or Social Media Apps (R11). This would create a gap between the system and VideoAR.

4.2 Geolocation and Mobile Related

In this section, more systems related to AR, geolocation and mobile technology are reviewed. All systems use Augmented Reality technology and some make use of geolocation technology. Compared to last section 4.1, none can play videos in an AR view.

The Touring Machine (Feiner, MacIntyre, Höllerer, & Webster, 1997) is one of the earliest AR prototypes that combines geolocation and mobile technology. It is built for touring the campus and offers immersive user experience. The system uses various equipment for proving the feasibility of the AR concept. A head-mounted, see-through display can display tour information by laying virtual text over the real buildings. A portable computer is put into a backpack for computing graphics and transferring data. Orientation tracker and GPS tracker are able to track and collect geo-information, for example, the location of users. Users communicate with the system via a stylus and trackpad. A handheld device running on a Windows 95 system, holds a web server inside. The webserver plays the role of communicating with the computer in the backpack. The webserver handles inputs from users and sends requests to the backpack computer to show graphics on head worn display. All communication is based on a local network. Power for all equipment is provided by a power belt worn by the user. The total weight is about 40 pounds, but the system shows the ability to move freely. The features are relatively simple. Users can see tour information floating over the buildings in the campus once they are looking at the architectures (R2 and R3). The information can be filtered by menu selections of the system (R9). The system also can guide users to find the desired tour spot according to their locations and orientations (R1). However, there are many tech issues in the system, but this is how the AR was utilised in the early days.

The system can navigate users in an AR view, though it fails to play a video (R4 and R5). It takes advantage of GPS sensors, but it cannot offer features such as Radar (R6) and Map (R7). Its 40 pounds combination of equipment can be impossible for tourists like Audrey to carry around (R8). It also lacks the ability to share and capturing data (R10 to R13) besides searching and browsing (Search and Browse POIs, R9). We review this work because it presents that how the AR technology was applied in early days. It

is a good example that highlights how modern AR technology is different. We do not expect this work to meet all our requirements.

MobiAR (Marimon et al., 2010) is a tourist guide information system that takes advantage of AR and geolocation technology on an Android mobile platform (R8). It helps tourists in delivering an enriched experience with multimedia information of outdoor and indoor tourist spots. In the study, Marimon et al. (2010) pointed out GPS on the mobile devices could cause a potential issue for placing AR objects over the real objects precisely, especially for the situation that two real objects are too close. This results from the accuracy of GPS. Their solution for the issue is combining computer vision technologies with geolocation data for calculating the correct location of the user. The system relays on a client-server structure. Firstly, a queried picture taken by the camera of the mobile is sent to the API of the server when the mobile is pointing at the real object (e.g. landmark). The server then processes the image query by a visual recognition engine which matches the picture with a set of geo-located images and give the coordinate of the user (R2). The server eventually returns the nearby Point of Interests (POIs) filtered by the user's preferences. MobiAR renders 2D and 3D graphics over the reality via its 3D visualization engine in terms of the information returned (R3). The information is listed in the system rather than being shown through the AR view (R4 and R5). Two scenarios are used for the evaluation of the application. In the former one, the user can point mobile devices at his surroundings. Then the POIs will be shown in the camera view (R1). The user can click POIs by touching the screen. Consequently, the system will list information (text or images) of them such as accommodation, restaurants, monuments and film-related videos. The information can be viewed by clicking the touch screen of the mobile devices (R9). In the latter scenario, a 3D model is shown in front of a landmark building for ease of understanding about the design concept of the building. The application also supports social features that users can post comments and pictures to the server in multi languages.

In comparison with our system, VideoAR, the MobiAR lists videos as tour information rather than embedding them in the AR view. It uses GPS technology to identify the ARPOIs. Nevertheless, it cannot offer Radar (R6) and Map (R7) features for users to be context-aware visually. It fails to share the location information to other applications such as Google Maps (R10) to help users to navigate in a broad range. MobiAR has the feature to create photos (R12) but lost the ability to record videos (R13) and communicate with other applications (R11).

AntarcticAR (Lee, Dunser, Nassani, & Billinghurst, 2013) is an Android application employs AR and mobile technology (R8). It allows users to take a virtual tour for Antarctica in a real environment without actually being there. Antarctica map is scaled down to a park size, then users can take fewer efforts and time to explore the adventure. The user experience is immersive but not as physically as HMD devices are, due to that users only can control the screen of a smartphone. Users can feel the Antarctica continent through four missions of the application. During the mission, they can gain tour information either by walking outdoors from a POI to the next one or interacting with the

POI markers on the electronic map by a touch screen indoors (R9). The tour information is represented by various formats such as stories (includes text and pictures), images, videos and panorama photos. However, none of them shown in the AR camera view directly during the tour (R4 and R5). Two navigation views are developed. The map view shows a map (R7) with the user's current location and a route to the destination. Users are tracked by GPS and the compass sensor for geo-located data. Another mode for navigation is AR view. In this mode, users can see through the camera and move without restrictions. The ground and trees are covered by snow (the semi-transparent white layer). The real soundings besides that remain the same, for example, the sky. The AR view keeps updating while users are walking. 3D virtual objects can be seen on the trip such as animated penguins, tents and huts. The POIs showed as 3D flags during the AR tour (R2 and R3). Users then can know where to walk to the next stop. The system presents a virtual radar (R6) which shows the next stop and a virtual character for assisting players. They are shown as needed. All tour data are stored locally on the mobile phone. Later, a survey for evaluation was conducted. Overall experience was positive. AR view mode was more fancied by outdoor participants compared to the indoor ones who enjoyed using the map view. The panorama pictures and video stranded out of four ways to present tour information. Ultimately, they stated that using outdoor AR application for a virtual tour is feasible and the user experience could be improved significantly.

Still, users are incapable to watch the video in the AR view of the AntarcticAR. Users can interact with POIs either by the touching screen or walking to the point. If they walk by the AR view, they are using the Navigate feature (R1). However, the location information (R10) cannot be passed to other applications. The lack of the social features (R11) and capture moments (R12 and R13) makes differences with our system. On the other hand, it provides similar features to our requirements in terms of Geolocation.

		Requirements													
		AR and Video						Geolocation		Mobile	Information management				
		Navigate in AR View	Identify the POIARs	Label the POIARs	Blending Videos into Reality			Dynamic Ratio	Radar	Map	Handheld Device	Search and Browse POIs	Share to Google Map	Share to Social Media	Capture Photos	Record Videos
Categories	**Systems**	R1	R2	R3	R4			R5	R6	R7	R8	R9	R10	R11	R12	R13
					Mode I	Mode II	Mode III									
Superimposing Video on an AR View	**Camera Operator AR**	✔	✔	✖	✖	✖	✖	✖	✖	✖	?	✖	✖	✖	✖	✖
	The Augmented Video Wall	✖	(✔)	✖	✖	✖	✖	(✔)	✖	✖	✔	✖	✖	✖	✖	✖
	Film-location-AR	✔	✔	✔	(✔)	✖	✖	✖	✖	✔	✔	✔	✖	✖	✔	✔
Geolocation and mobile related	**The Touring Machine**	✔	✔	✔	✖	✖	✖	✖	✖	✖	✖	✔	✖	✖	✖	✖
	MobiAR	✔	✔	✔	✖	✖	✖	✖	✖	✖	✔	✔	✖	✖	✔	✖
	AntarcticAR	✔	✔	✔	✖	✖	✖	✖	✔	✔	✔	✖	✖	✖	✖	✖

Table 2: Comparison of related work. ✔means fully support; (✔) means partially support; ✖ Means not support; ? means unknown support or not.

4.3 Summary

Table 2 shows a summary of the analysis of related systems. Different marks are used for clarifying how a system supports a requirement. The tick mark, ✓, represents that a system fully supports the requirement and a mark with parentheses, (✓), means the system partially supports a requirement. The cross mark, ✖, signifies a requirement is not supported by the system. The question mark, ?, indicates that it is unclear from the literature if a system supports the requirement.

From the AR and Video category (shown in the grey colour in Table 2), we can see that all systems fail to fully support R4 and R5. Only the Film-location-AR (Park & Woo, 2015) system partially supports one (Mode I, Video Overlay) out of three modes in R4.

The Geolocation category (shown in the blue colour) in Table 2 shows that only one system supports for Radar feature (R6). Similarly, only Film-location-AR and AntarcticAR (Lee, Dunser, Nassani, & Billinghurst, 2013) support the Map feature (R7) as they allow users to follow routes on the map in a broad range. However, the requirements in this group are not considered to be a major challenge in this research.

As for Mobile category in Table 2 (as the column in the pink colour shows), the requirement R8 is supported by four systems. All these systems are running on Android mobile phones. The equipment used in The Touring Machine (Feiner, MacIntyre, Höllerer, & Webster, 1997) is too heavy to be carried for long-distance trips. The status of Camera Operator is current unclear.

The last category is Information Management shown in the green colour in Table 2. In this category, R10 and R11 are not supported by any system in the table. However, these features should not be barriers for these systems if there are needs for them because they can be easy to develop on a mobile platform nowadays. Also, they are not our focus, although they are not covered by the related systems. Additionally, the requirements in this category are more for entertaining users instead of solving a technical issue.

By reviewing these systems, a pattern emerges in Table 2. Some easier-to-implement features (e.g., R10 and R11) are not available in all systems. Our focus is on the challenges posed by AR and Video category (R1 to R5). The features (R1 to R3) are covered by most of the systems we reviewed. Therefore, they are not the challenges from a technical perspective. None of the analysed related works fully cover the feature (R4) that we envisioned, and only one system partially addresses the Dynamic Ratio (R5) feature. Hence, these requirements (R4 and R5) will be the central focus of this thesis.

5 Design and Implementation Considerations

In this chapter, we illustrate what and how development resources are selected for the application and the design and implementation of our application.

Before the implementation, an adequate consideration for development tools is required, particularly in light of the requirements identified in Chapter 3. We first explore what operating systems and devices are suitable for the research goal. Then, several frameworks and libraries are explored and compared in terms of fitting our scenarios. Finally, the rationales for selection of these candidates are also clarified based on the preferences of our application.

5.1 Mobile Operating Systems and Devices

The VideoAR prototype system is designed for mobile phones or tablets. Two platforms can run AR applications: Android and iOS. The native programming language is JAVA for Android and SWIFT or Object-C for iOS respectively. We compare these two operating systems and their devices according to software distribution, cost of the development, market share and available devices.

Software distribution: Android is better than iOS when using only for the research purpose. Users can install Android package files directly on their phones or from the official application store of Google (Google Play Store). For iOS, users only are allowed to download applications from Apple App Store.

Cost of the development: The development on the Android platform relatively more affordable than on iOS. To develop an iOS application, 99 USD member fee is required. The development of Android is free if there is no need to release the application on Google Play Store where the public can download applications.

Market share: According to (International Data Corporation, 2017), Android phones are more popular than iPhones. Android phones take up 83.4 percent of the smartphone world market compared to 15.4 percent by iOS phones.

Available devices: Android phones come from various manufacturers. The Nexus series of smartphones have better compatibility than others with the Android platform, this is because it is designed and made by Google which made the Android system as well. iOS devices (iPhones) only can be made by the Apple company. They both have quality cameras and sensors such as compass sensors, gyroscope sensors and GPS sensors.

Depth sensors: Depth sensors can be helpful for Dynamic Ratio (R5). This sensor can measure the distance between the device and the real objects. Therefore, it is easier to keep right ratio of the video to the real objects by calculating the distance. This is optional for our research since the common phones do not have this camera built in.

5.2 Frameworks

For the software development, we have to decide if we wish to use existing AR libraries or develop a native Android application. In this thesis, a framework can be defined as a

development kit that allows users to build an application running on one or different operating systems by using its APIs. These frameworks often have their own or support third-party Integrated Development Environments (IDE) and programming languages requirements. When using these frameworks, such as Android[2], iOS[3], Unity[4], etc., the application is created within an IDE by calling and following the functions and procedures of the framework, then compiled into programs running on different operating systems. For example, Unity (Unity, 2017) is often used for developing mobile games, AR and Visual Reality(VR) applications. For creating an AR android mobile application using Unity, the developer writes C# code to call APIs of the Unity framework and then exports the program into applications by Unity IDE running on operating systems such as Android, iOS, Mac OS, Windows, etc.

Frameworks can be divided into two categories. They are the native framework and the third-party framework. Each of them has its own strengths and weaknesses. Native frameworks can be the frameworks that come with the operating systems. It has the best compatibility and performance with its operating system. It provides rich APIs for developers to build applications. However, it only supports its own operating system. Since native frameworks are often bound with the operating systems, they are more likely to have more users and external developers (developing applications for the operating systems) than the third parties. The solutions to common development issues on these popular platforms are well explored by the developers. It will be straightforward to find a solution to the issue we might meet. These factors are important when developing advanced features like collecting data from sensors and solving technical issues. They can reduce the time of development significantly.

Third-party frameworks often provide more features that native framework does not offer directly such as AR, geo-map and video players. Third-party frameworks are cross-platform in general. They can be time-saving due to that developers only need to deploy features one time rather than doing for different operating systems. Because of this, they have to use APIs of native framework offered by various operating systems for compatibility. Some strategies are often applied, for instance, translating their code into native code or running as a web application. They are middleware between developers and native APIs. This also can bring obstacles for developers to understand the principle of frameworks if there is something wrong when developing. Developers cannot really see how these frameworks communicate with the native API. Due to the same factor, the implementation of advanced customised features may be limited to a certain degree. Developers only can use the API of the native framework through third-party frameworks. They cannot use more advanced APIs from the system, for example, control the sensors directly. The third-party frameworks often cause poorer performance and lack

[2] Android, a mobile system with rich features, https://www.android.com
[3] iOS, an operating system that can run on mobiles. https://www.apple.com/nz/ios/ios-11/
[4] Unity, a popular game engine that is cross-platform. https://unity3d.com

of ability to use native interfaces or API. This is because they are complex and trying to be compatible with many operating systems.

Two native frameworks, Android and iOS, are explored. Three third-party frameworks are also inspected since they have different focuses, for instance, Unity emphasises on games and graphic development; Apache Cordova is using web technologies to simplify development crossed operating systems; Titanium allows developers to have near-native performance and build cross-platform applications.

- *Android framework*[5] is a native framework offered by the Android operating system. It has the best compatibility for the Android OS. The IDE Tool is Android Studio and the programming language is JAVA. It offers system-level APIs to enable developers to manipulate sensors on the devices. Other frameworks have to use these APIs as well if they want to be compatible with the Android operating system.
- Similarly, *iOS framework*[6] is running natively on iOS operating system developed by Apple. The native programming language is Object-C or SWIFT. The IDE tool is Xcode. It has rich APIs for developers to develop iOS applications. It is only compatible with the devices from Apple.
- *Unity*[7] is a popular development framework for games, AR and VR applications. It supports multiple operating systems (such as Windows, Mac, Linux, Android, Xbox 360) and can even run within a web browser independent of the operating system. The programming languages for Unity are C# and UnityScript. The devices that can run Unity code are various, for example, mobile phones, head amount devices, TVs and PCs.
- *Apache Cordova*[8] is a mobile development framework based on web technologies. It allows users to build applications that are crossed platforms. Users can code in web programming languages such as JavaScript, HTML 5 and CSS3. It has no specific IDE tools to offer as developers can choose any editor that is compatible with web programming languages. The frameworks support mobile operating systems (Android, Blackberry, Windows and iOS) and desktop platforms (Windows, Linux, Mac OS). The wide devices are also supported according to these operating systems. However, developers cannot use native interfaces. The performance of Apache Cordova may be poorer than using native frameworks since it basically works like a web tab in a browser on a mobile phone.
- *Xamarin*[9] is a framework focus on offering native user interfaces, API access and performance while being cross-platform. It is using C# as the programming

[5] Android, a mobile system with rich features, https://www.android.com
[6] iOS, an operating system that can run on mobiles. https://www.apple.com/nz/ios/ios-11/
[7] Unity, a popular game engine that is cross-platform. https://unity3d.com
[8] Apache Cordova, https://cordova.apache.org
[9] Xamarin, https://www.xamarin.com

language and Visual Studio as the IDE tool. It supports operating mobile systems such as iOS, Android Windows Mobile. It takes advantages of that iOS and Android can call the code compiled directly if the code is following the standard, for example, iOS can call assembly language (ARM) code and Android calls the code compiled by just-in-time (JIT) compiler. The framework uses Mono[10] library to deliver the code that these operating systems can use. However, how the framework translates its code into the native code is unknown.

5.3 Libraries

Library is a set of functions that extend the functionalities of a framework. Libraries can cooperate with an application developed by a framework, while the main application is written in native code of the framework. They can be written in native code or native compatible code. For example, the Vuforia for Android[11] library is used in Android applications to incorporate AR features based on the Android framework on an Android operating system. The drawback is that most libraries are not multi-platform compatible.

Since there are many libraries currently, there are several criteria set for filtering suitable libraries. The most important one is the library selected should be able to play videos in AR on mobiles phones (Mode I in R4). It also can play videos with a transparent background in order to fulfil the Mode III (Video Background removal) in R4. For ease of developing and better performance, it is better that the library chosen is written as a native library. The optional criteria are that the library can be highly customizable and well maintained. This is for the subsequent development in the future.

We explored potentially suitable libraries for the use in our project. These libraries are discussed in aspects such as playing videos in AR with or without transparent background, Android compatibility, Android native library, customizability and update frequency.

- *ARCore*[12] is a native library from Google. It has the best capability and stability for Android as they are developed by the same company, Google. It supports frameworks such as Android, Unity and Unreal. It offers the source code online[13] and it is still updated by Google. However, it is unknown if it can play video in AR. It required the latest Android devices like Google Pixel or newer model.
- *Vuforia* is a library that enables playing videos in AR and provides native libraries for Android and iOS (Vuforia, 2017). It has difficulty to play videos in an alpha channel (transparent background). The library is compiled into a package that can be called by the Android framework. The developer cannot see the

[10] Mono, http://www.mono-project.com
[11] Vuforia for Android, https://library.vuforia.com/articles/Solution/Getting-Started-with-Vuforia-for-Android-Development.html
[12] ARCore, https://developers.google.com/ar/
[13] arcore-android-sdk, https://github.com/google-ar/arcore-android-sdk/releases

source code in this case, so the library cannot be modified by the third-parties. It still releases new versions and well maintained.

- *Wikitude*[14] is a library that enables playing videos and transparent background videos in an AR view (Wikitude, 2017). However, only the library written in JavaScript support these features. The native library of Wikitude does not support playing videos in AR. All versions of the library are Android friendly but not able to be modified since it is compiled into a .so file. It is well maintained still now.
- *EasyAR*[15] supports Video playing both with transparent and none-transparent backgrounds. It has native SDK for Android. It has no issues to support Android devices. It has rich features and is easy to configure, although the development documents are relatively poor. It is well developing still. To play videos with a transparent background, it has the requirement for the videos. The videos are required to have the RGB channel and alpha channel with the same content on the left side and right side respectively. This is quite different with the common videos with a transparent background (green background).
- *ARToolkit*[16] supports video playing both with transparent and normal one. It is well running on Android devices and offers Android native library. It is an open source library, this enables developers can produce a highly customised application and the principle of the library is visible for the public. The developer can fix the issue on a library level when they encounter an issue, but this is time-consuming and difficult as it is complex. The library is updating slowly and has many bugs unfixed. The examples of it cannot run on Android studio (Android development environment) 2.3.3 or over. It is only compatible with JDK 1.7 which is rather old, compared to the latest one 1.8. Additionally, JDK 1.8 is required by the latest Android studio.
- *MAXST*[17] is another framework that videos including transparent ones can be rendered over the real objects. It is well maintained still. The compatibility with Android devices and framework is clearly supported. Similarly, it is sealed into a file that can be called by the Android framework. Therefore, developers fail to customize it. It is well maintained so far.

5.4 Summary

In this section, we made the decision about devices, mobile operating system, framework and library.

The Android phones were chosen for this research as it is easier to distribute the application and the cost is relatively low for development. The Nexus series of Google is

[14] Wikitude, https://www.wikitude.com
[15] EasyAR, https://www.easyar.com
[16] ARToolKit, https://artoolkit.org
[17] MAXST, http://maxst.com

appropriate compared with other brands of Android devices since its operating system and hardware both developed by Google. The phone must run the Android platform (version above 4.3), in order to use most of AR frameworks and libraries listed in Section 5.1.2.

Due to the factor that this thesis is carried out for research purpose, more highly customised features are likely to be used. Since the Android device has been chosen as our option, the Android framework, therefore, is selected as the development framework.

	ARCore	Vuforia	Wikitude	EazyAR	ARToolkit	MAXST
* Play videos in AR on mobiles	?	✔	✔	✔	✔	✔
* Play videos with transparent background.	?	✔	✔	✔	✔	✔
* Android compatible	✔	✔	✔	✔	✔	✔
* Android native libratry	✔	✔	✖	✖	✔	✔
- Library customizable	✖	✖	✖	✖	✔	✖
- Update frequently	✔	✔	✔	✔	✖	✔

Table 3: Frameworks and libraries. ✔means fully support; (✔) means partially support; ✖ Means not support; ? means unknown support or not. * means required criteria; - means optional criteria.

We compared all libraries in Section 5.3 as Table 3 shows. ARToolkit and MAXST meet all required criteria we set. Besides these two, the other libraries cannot meet our required criteria. Therefore, they are removed from our candidates list. According to our test, the current version of ARToolkit has issues to run on the latest Android studio (Android development environment) and has many bugs when playing videos. The code itself can be complex and hard to understand. MAXST can play videos with and without a transparent background smoothly. The code of MAXST is relatively easy to understand and to be integrated. After considering these factors, MAXST library for Android is suitable for our goals.

6 Prototype Implementation

In this section, we draw the experience from how those popular libraries in Section 5.3 by analysing their sample codes. Then we discuss development methods for best addressing the key requirements identified in Chapter 3, i.e. Blending Videos into Reality (R4) and Dynamic Ratio (R5). There are three modes to play videos in AR in the requirement 4 (R4). Mode I (Video Overlay) can lay regular videos over real objects. Compared to this, Mode II (Video Reshape) not only can play videos in AR but also can shape videos to match the shape of the object that videos are superimposed on. To completely blend videos into reality, Mode III (Video Background Removal) can remove the background of the videos and embed them into reality. Another requirement R5 enables videos to keep the right ratio to the real objects. In other words, the size of the video in AR can be updating constantly according to the size of the real object. We also explore the object tracking technology to extend Mode III.

Both common videos and customised videos are used in this section because we cannot use movie original clips from the movie, The Lord of the Rings, due to copyright issues (seen in Figure 14). We filmed a video which has similar content to the movie clip (As Figure 13 shows). This video is used throughout the tests as an alternative for the original movie clip. All photos or videos are taken by the author not explicitly stated otherwise. The character in the videos is the author.

Figure 13: A man is knocking the door

Figure 14: Gandalf knocking the door of Bag End (Jackson, Osborne & Walsh (Producer), & Jackson (Director), 2001)

6.1 The Method to Play Videos by These Libraries

Section 5.3 introduced numerous libraries; here we discuss how they support the playing of videos. We analysed code examples offered by these libraries (MAXST, Vuforia and EazyAR) and draw conclusions on how the libraries work for playing videos in AR. They all share similar principles, as follows. ARCore was not tested because it requires the latest mobile devices which currently are unavailable to us.

The libraries will get control of videos captured by a camera first. Then they draw shapes over the targets with the texture loaded with a video using Open Graphics Library (OpenGL[18]). The library is known as OpenGL ES in the Android framework. OpenGL is a graphical API of hardware that running on multiple platforms. It can process 2D and 3D graphics. OpenGL ES can lay 2D or 3D graphics with a video texture on, so as to achieve the goal that overlaying videos in AR. OpenGL also can modify the colour of the texture. So, the videos can be shown with the transparent background. For example, we can filter the green colour and make it to be transparent when displaying a green screen video. The part that is not green will be shown as it is. This is also known as Chroma Key technology.

Next, we describe how the five main features were implemented for our prototype, using the selected library.

6.2 Implementation of Mode I: Video Overlay

MAXST AR SDK is chosen as the library for this research in Section 5.4. It has the Image Tracker feature that can achieve the Mode I, Video Overlay in R4. The feature

[18] OpenGL (Open Graphics Library), a library for developing 2D and 3D graphics applications, https://www.opengl.org

can recognize targets that are uploaded to the Target Manger, then cover the target with an image or video. Target Manager is a tool from MAXST AR SDK to manage AR targets. These targets are pictures used for the identifying purpose. We first need to upload pictures of the object to be identified to the Target Manager, then download and load the related databases from Target Manager into our application by the Android framework. The database is made by Target Manager according to the target pictures we uploaded. Next step is linking the video to be played with the targets. By using the libVideoPlayer.so library offered by MAXST, we can create an AR video player and load the videos. Then we use OpenGL ES library[19] to draw this video player as a three-dimensional model over the target. This OpenGL ES library is built in Android system.

Figure 15: Video overlay result in the Hamilton Gardens

Our application was tested in Hamilton Gardens as Figure 15 shows. The left side of Figure 15 is a sign with Chinese characters on the wall of the Chinese Scholar Garden in Hamilton Gardens, which is used as a target to be identified. The right side of Figure 15 indicates the result that laying a video clip over the sign. The shape of the video appears to be rectangular by default.

The R5 (Dynamic Ratio) in Section 3.1.5 is also fulfilled as Figure 16 shows. The left side of Figure 16 is what the user can see when the user is in a close distance (less than

[19] OpenGL ES, the Android version of OpenGL, https://developer.android.com/guide/topics/graphics/opengl

about one meter according to our measurement) from the target. The right side of Figure 16 shows that the target is still stable when viewing it from a further distance, approximately less than three meters. However, once the user steps back more than three meters from the target, the video in an AR view would start to shake and even disappear. The distance for holding targets by MAXST AR SDK is unclear in official documents. These numbers about distances are measured by the author. During walking from a further distance to a closer distance, the target zoomed out and kept the right ratio to the target. This fulfils the requirement 5 (R5).

To extend Mode I, Video Overlay in R4, watching videos from side angle was tested and works stably. Watching videos in an AR view from an extreme angle like 180 degrees or zero degree fails to hold the video on the target according to our tests. The testing result is as Figure 17 shows. The video is stable and clear when the distance is in an effective range.

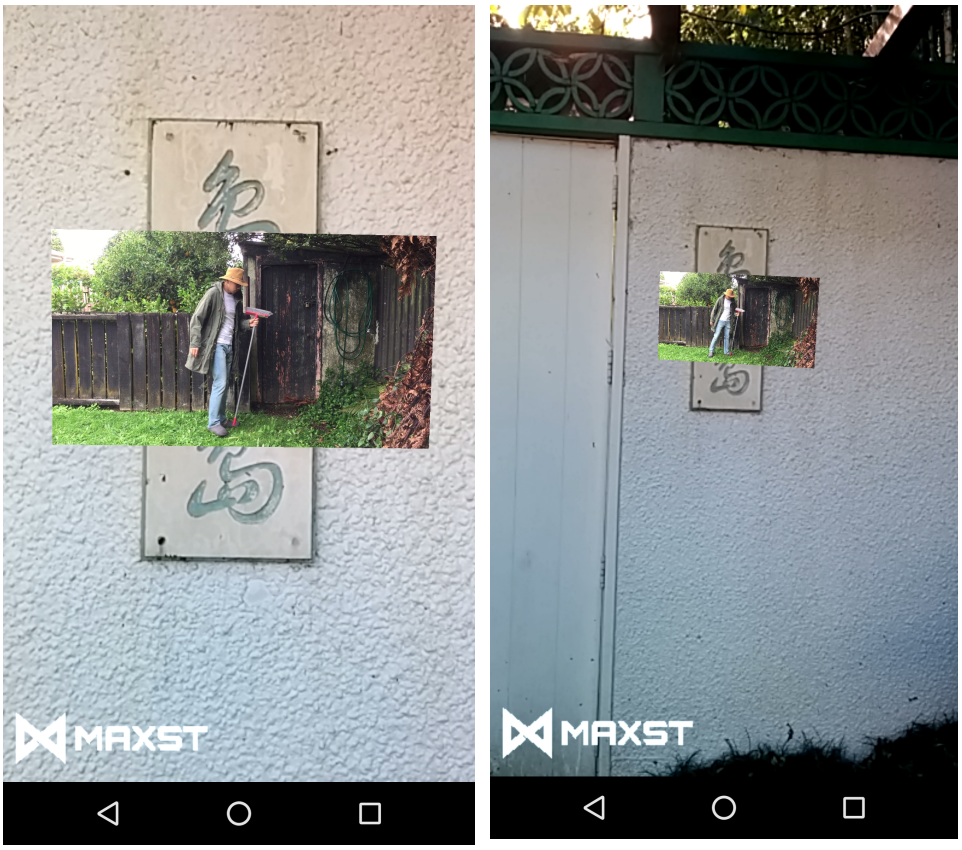


Figure 16: Dynamic ratio of the video

Overall, the scenario in Step D of Section 2.2.1 is fully deployed by our application. The user can view a target from a close or far distance and even from the side angles. However, the distance is limited down to less than 3 meters and the extreme angles are not covered by the application. To fully develop the R4, Blending Video into Reality, more efforts in terms of Mode II Video Reshape is shown in the following Section 6.3.

Figure 17: Watch videos from the side angle

6.3 Implementation of Mode II: Video Reshape

After overlaying videos in AR in the last section, it is time to take a further look at how to draw videos with different shapes in an AR view in this section.

Before drawing shapes, two things need to be considered. The first one is how to draw a shape that videos can be attached to in an AR view. Secondly, how to draw a 2D model in 3D space. In AR, everything should be three-dimensional. By taking advantage of OpenGL Library, 3D models can be rendered and loaded the video player from MAXST AR SDK as a texture in the camera view. The way to render a 3D model is using OpenGL to draw shapes by three-dimensional coordinates. By changing coordinates, the different shapes with textures can then be drawn. The video usually is played as a 2D shape rather than a 3D model, for example, the rectangular shape (as Figure 14 shows). In order to achieve this, the third dimension of the model needs to be limited to zero.

In this section, we use a rectangular, a solid circle and a cube as examples to represent the various shapes that can be drawn. The rectangular is the basic shape for videos; the

solid circle shape is the one with curves and the cube is a 3D model which differs from the 2D shapes like the rectangular and the solid circle shapes.

6.3.1 Rectangular Shape

The rectangular shape is the default shape of the video. In Mode I: Video Overlay, we play a video in this shape (see Section 6.2). To draw a rectangular video in an AR view, a 3D model needs to be created first. As we are drawing a 2D rectangular shape as a 3D model in a 3D dimensional space, the third dimension should be always zero. The dimensions of the models are manually calculated in this case as it is a simple shape, then Open GL can draw the rectangular according to the dimensions. To understand what we did here, imagine that there is a zero-thick rectangular paper floating above a book, the book is the target in the AR view. The paper is a 2D model. It covers the target on the top and loads the video texture (video player) on its surface. When using three-dimensional coordinates (x, y, z) to represent the zero-thick paper, the z coordinate is always empty. Open GL can draw different shapes by using triangles, which means a square can consist of two triangles as Figure 18 shows. Each vertex of triangles has a three-dimensional coordinate. Two triangles share the same edge. The actual effect can be seen in Figures 15 to 17.

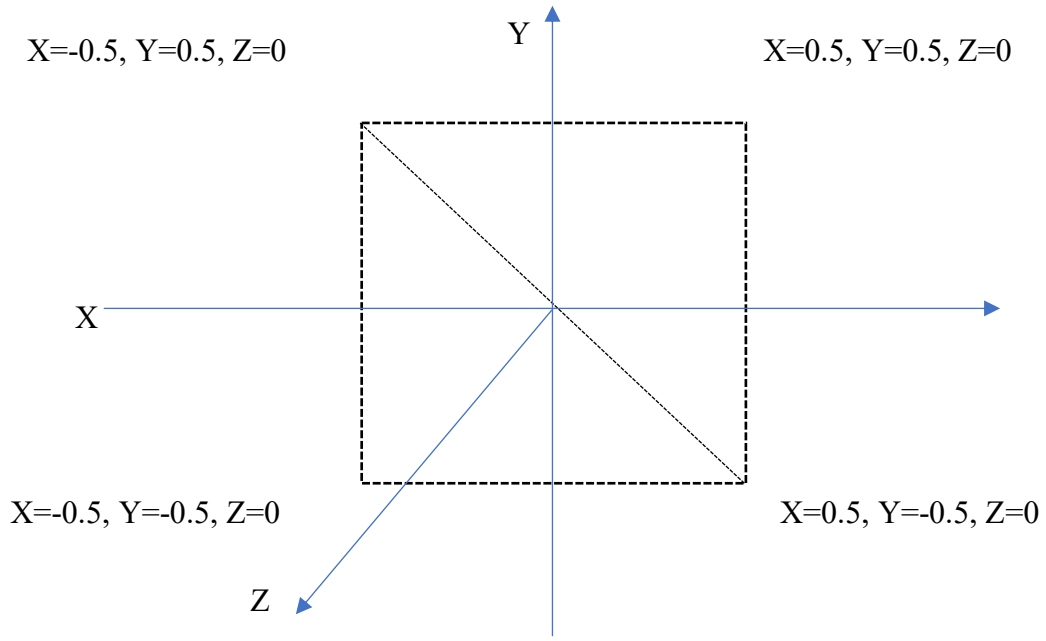


Figure 18: Rectangular shape.

6.3.2 Solid Circle Shape

A solid circle shape similarly can be drawn in an AR view over the target. This helps with the Mode II, Video Reshape in R4. The door in Figure 11, R4 is round. If we could cover the door with the similar shape video, then the video can much more fit in the

environment than the rectangular shape. The excessive parts (corners of the rectangular shape) of videos also can be removed by solid circle shape.

Since the circle is round and solid, we then need to draw it by using triangle fans instead of triangles used for the rectangular shape. A circle shape can be made of several triangle fans. Still, we need to create a 3D model first and makes z coordinate of each vertex of triangle fans to be zero, then it can cover the target with zero thickness. In Figure 19, there are 8 triangles fans. Some of them are sharing the same edges. The more triangle fans the solid circle consists of, the more round the solid circle can be.

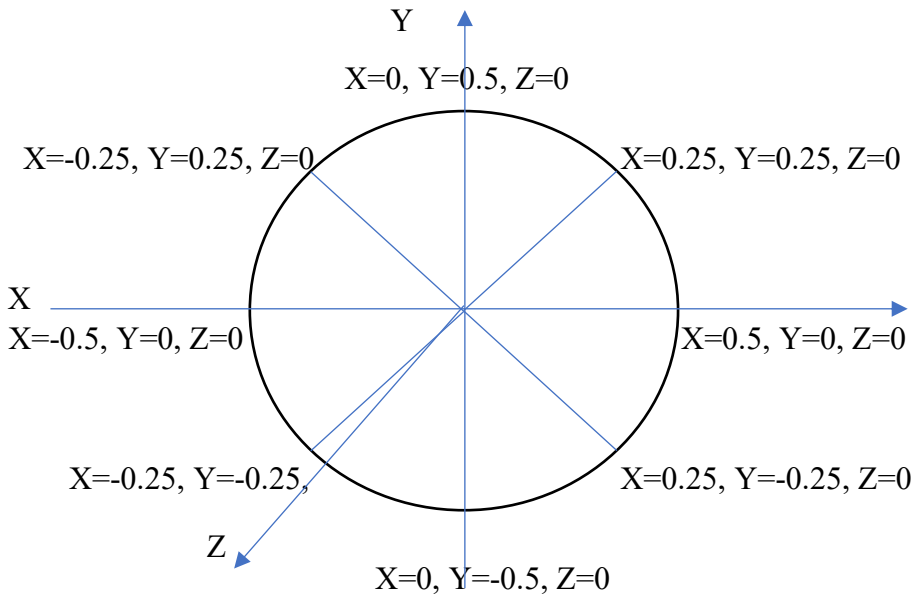


Figure 19: Solid circle shape

Our application was used for drawing a solid cycle shape. A video is filmed in the Hamilton Gardens which includes a round door. It is used for the video laying over the target. Figure 20 shows the testing result. The left side of Figure 20 is a round cap of a bottle. It is chosen because its shape is round, then we can cover it with a round shape video. The right side of Figure 20 shows a round shape video laying over the same cap. From a programming perspective, drawing shapes with curves is much more complex than drawing lines, especially in calculating the three-dimensional coordinates. A software, Blender[20], is used for calculating the coordinates of the round shape loaded with the texture of a video. Blender can draw 3D shapes and export data to a waterfront file. A waterfront file is a format of data for geometry. By analysing the file, we capture the coordinates and input into our application. The application now can draw a solid circle according to these coordinates. However, the shape starts spinning or vanishing if the user is too far away from the target. This could be improved if the user walks closer to the target. Another issue is the round door in the video is misshaped slightly. This could

[20] Blender is a free and open source 3D model design tool. https://www.blender.org

be solved to change three-dimensional coordinates. The reason is that where and how much to cut the video is highly based on the coordinates built in the application. The test result above has proved that solid circle shape video is possible for rendering in the AR view. This is potentially useful for drawing shapes with curves over the items with similar shapes, so creates a better user experience for blending visual items into reality.

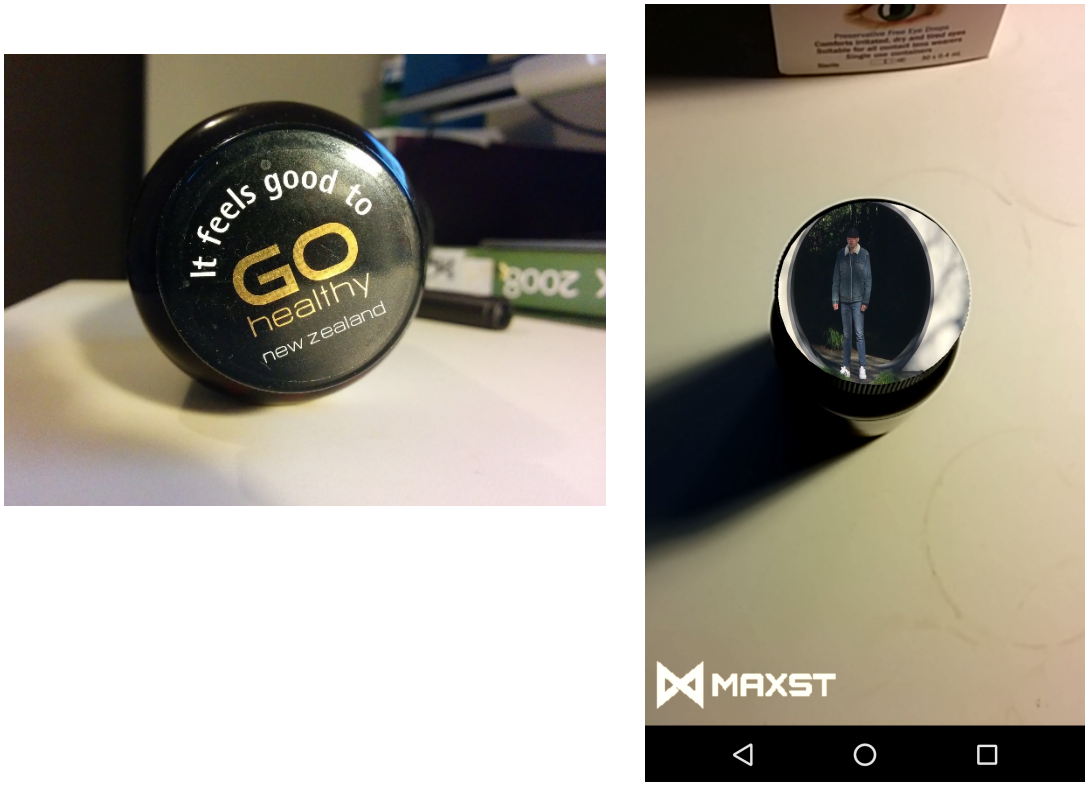


Figure 20: Solid circle shape drawing

6.3.3 Cube Shape

The cube shape is a 3D shape which is different from the rectangular and the solid circle shape we drew previously. We will use this shape to solve the issue that watching videos from a side angle in R4. Then users can watch the video from any side of the cube except the side overlapped with the target.

The 3D cube is still can be rendered by triangles by Open GL library, if we think it has six sides and each side has two triangles. The Z coordinate has its value since it is no longer a 2D model. In Figure 21, we can see that the side with red borders can be divided into two triangles and they share the same edge.

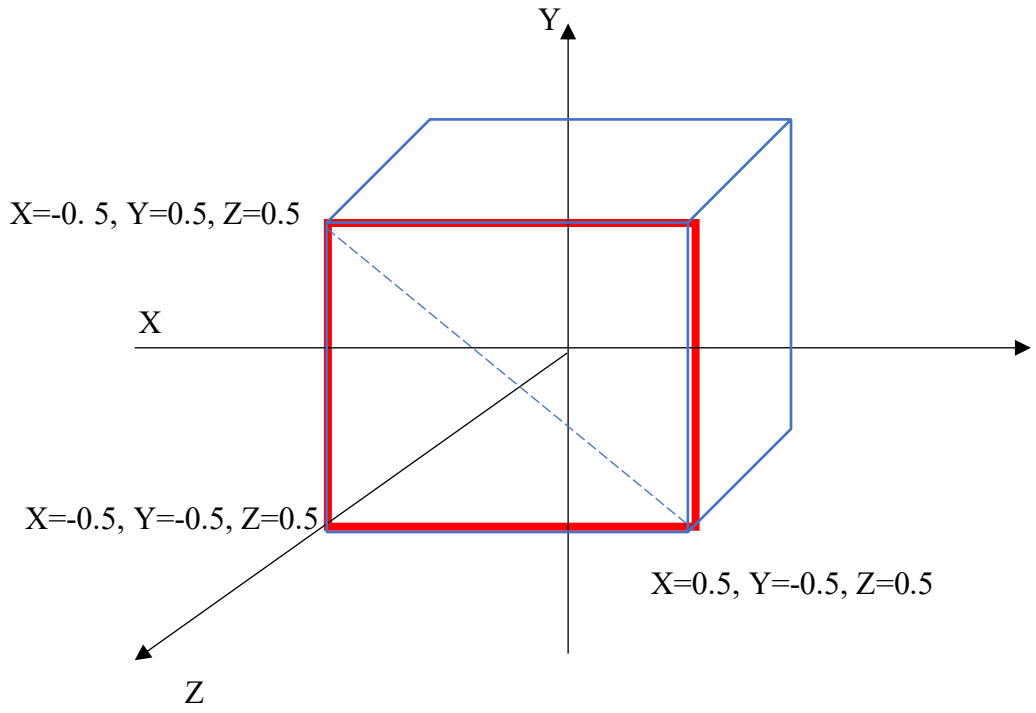


Figure 21: Cube shape

The testing (as Figure 22 displays) for the cube shape video in Hamilton Gardens shows that the result is not good. It can solve the issue that watching the video from a side angle to some extent. However, the MAXST AR SDK limits the ability of holding and recognising from a very extreme angle. The user experience of watching the video is not as good as the rectangular shape video (Seen in Figure 16). Shaken, vanished target and other issues like spining[21] occurred as we mentioned in Section 6.2 Implementation of Mode I, Video Overlay. These issues can be addressed by walking closer to the target or viewing it from a normal angle, for example, from 30 to 150 degrees. The characters in videos in Figure 22 are upside down on some sides, this can be corrected by changing the coordinates.

[21] The new issue that specifically for the cube shape movie emerged. In Figure 23, we can see that if viewing the target from a very side angle, the cube would start to spin. However, the exact reason for this phenomenon is hard to determine because the library is presented as a black box. The best guess is the MAXST AR SDK cannot constantly recognize the target and lock the position in an AR view in this case. It results in the issues such as shaking, vanishing and spinning.

Figure 22: Playing the cube shape video

Figure 23: Spinning cube when viewing it from an extreme angle

6.4 Implementation of Mode III: Video Background Removal

In this section, we explore how to remove the background of videos. The MAXST library as other libraries (see Section 5.3) has the capacity to play videos with the green colour background and make the background of videos to be transparent. After reading the code, we found the real reason that the library can play video with a transparent background is that it takes advantage of the feature, Blending of OpenGL. This feature can filter and sort the colour of pixels and blend the pixels back to the original graphic. This feature also makes the pixels transparency. The threshold can be set by programming (OpenGL, 2017). However, the green colour range applied in this research is (0, 0, N>0.2) in RGB colour. Each value of RGB is from 0.0 to 1.0. For example, the red colour is (1, 0, 0) and the green colour is (0,1,0). Then any green that is more than 0.2 is considered as the appropriate green to be converted into transparency.

This technique is especially useful for the Mode III – Video Background Removal in R4 (see Chapter 3.1.4). The green screen video that is a video with the green colour background. This type of video is widely used in the movie industry, especially in science fiction movies. It allows technicians to separate characters from the background if the background is in the vivid green colour. Then technicians can replace the green part with spaceships or other unrealistic things that do not exist. To covert a normal video into a green screen video, Background Subtraction technique is needed.

Video Background Subtraction is a widely used technique in computer vision systems. The basic idea of Background Subtraction is separating moving objects (the foreground) from the static images (the background). The work of Bouwmans (2012) summarises the four processes of the common background subtraction as follows:

- **Background Modelling**: A model to represent a background is built on this stage.
- **Foreground Modelling**: A model to detect moving objects as the foreground and compare each pixel with the background for classifying that the pixel is from the foreground or the background.
- **Background Maintenance**: The background is updated according to the foreground detection, for example, anything that does not move for a long time will be integrated into the background. Light and shadow changes also can be processed during the stage.
- **Post-Processing:** A mask that covers the background can be created for highlighting the foreground (i.e. moving objects).

OpenCV[22] (Open Source Computer Vision Library) is used for converting a non-green background video to a green video in this research. This library can detect, track and recognize objects in videos or pictures. It is open source, and has more than 2500 algorithms built in. It implements the Gaussian Mixture-based Background/Foreground Segmentation Algorithm based on the works of Zivkovic (2004) and Zivkovic and Van Der Heijden (2006). The algorithm enables OpenCV to detect shadow too (OpenCV, 2017).

[22] OpenCV, https://opencv.org/about.html

Figure 24: Covert normal videos to green screen videos.

We built a software written in C++ that can separate the foreground from the background via OpenCV. It can change the colour of the mask into green at the step of the Post-Processing stage of Background Subtraction. The result is shown in Figure 24. The video is processed and the background of it is replaced with the green colour. The green part is considered as the background since the objects in it are not moving. The moving person (the character from the video) remains colourful. However, background subtraction is still not accurate. We tested our software in Hamilton Gardens as Figure 25 shows. We can see that the background recognition is not accurate, a part of the foreground (The person in green coat) is removed accidentally.

Figure 25: A Processed video in AR

Another way to remove a background from videos that we explored is using a pre-filmed green screen video. This type of videos is filmed with the green background. Or we can say it only has the foreground which is anything we consider is necessary to show. We did not have access to the facilities to film a full-size green-screen video for this project and, therefore, use throughout this chapter freely available examples (Figures 26 and 27) and smaller-size green-screen videos (Figures 28 to 32).

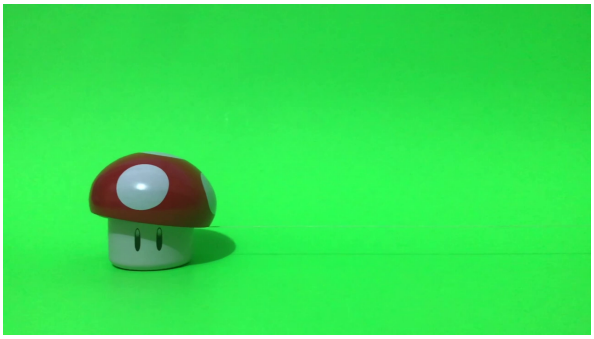
In Figure 26, we can see a pre-filmed video with the green background. Figure 27 presents the result that the video blended into Hamilton Gardens. It can be better compared to Figure 25 in terms of the effect. However, the drawback is pre-filmed green screen video is customised. The movie clips or common video normally do not have green screen background.

Figure 26: A man talking on the phone against green screen (Royalty Free Stock Footage, 2016, May 4)

Figure 27: Pre-filmed green screen video, a man taking a call in Hamilton Gardens. (Royalty Free Stock Footage, 2016, May 4)

We managed to film green screen videos for the application to utilise. A green paper board (50cm width and 65cm length) is used as a green background of videos. Toys with different reflectance properties are used as filming objects indoors and outdoors. In a room with a white light lamp, we used a mushroom toy to move around in the front of the green paper board as the left side of Figure 28 displays. Compared with this, the right side of the figure shows the actual effect when using the footage in the AR view. The bottom part of the mushroom toy is wiped with the green background together. This is due to the reflection factor. The surface of the toy is highly reflective and reflects the green colour.

We then tried to explore the effect for using a rough surface toy, Stitch toy, as the left side of Figure 29 shows. The Stitch toy reflects less light than the mushroom toy overall. It shows an almost complete image in the AR view as the right side of Figure 29 illustrates. However, there are some parts of the toy that are still cut out accidentally, for example, the bottom of the Stitch toy. After this, a pure pink colour paper slip is used because the surface of the paper is the less reflective compared to the toys. It is clear to see that in Figure 30; the pink paper remains intact except the part covered by the fingers. This proves the reflection can be an issue for filming the screen green issue. The way to solve this is making the item to be filmed less reflective.



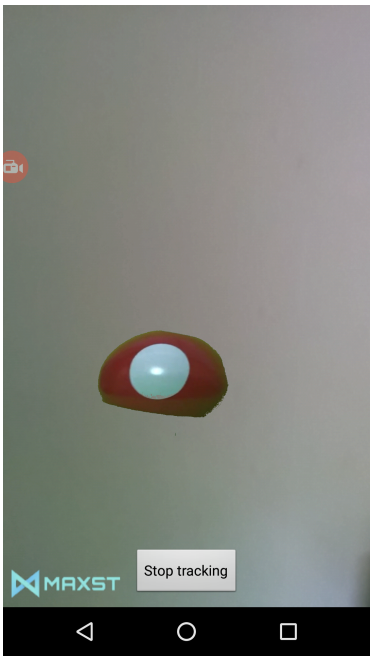


Figure 28: Mushroom moving

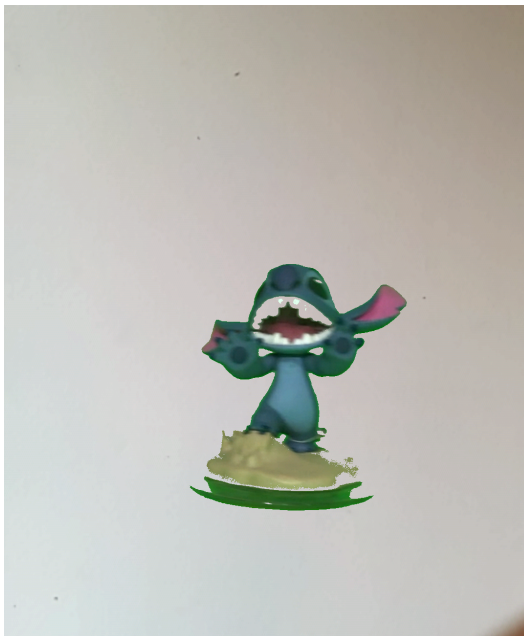


Figure 29: Stitch toy test

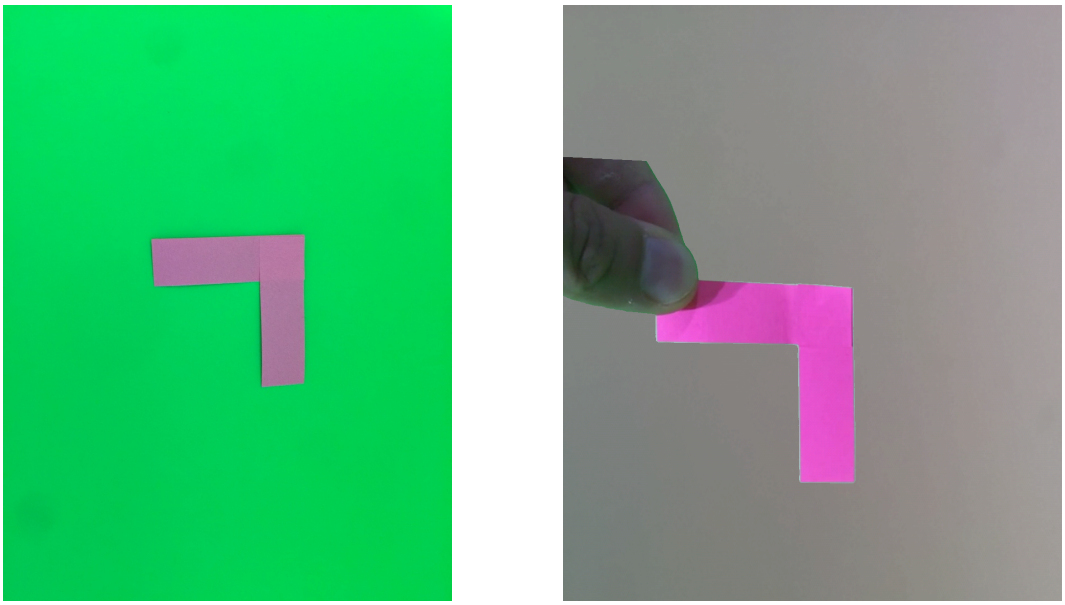


Figure 30: Pink paper test

Besides filmed indoors, an experiment did outdoors as well. Figure 31 shows two red toy cars. The left side one was filmed indoors under a white light lamp; the right one was filmed outdoors by using natural light. We can see that the left one has more reflections, especially on the wheels. Two cars are projected on a door by our application as Figure 32 shows. The top one (filmed indoors) of Figure 32 has more parts to be removed compared to the bottom one which is filmed under the sun light. The natural light enables the camera to capture more details while generating fewer reflections. The indoor lamp is easy to concentrate too much light on a spot so as to confuse the camera and miss the details.

By these experiments, we conclude that reflection can disturb green screen videos when using them in the AR View. Choosing items with appropriate reflectance and light source when filming green screen videos can improve the issue. Items with a rough surface can work well to some extent. Sufficient indirect light also can create less reflection. For example, a photographic softbox can diffuse the light and project adequate light. Unfortunately, the softbox equipment is unavailable and not tested in this research. The device used for filming is an iPhone 7 Plus. The more advanced cameras may improve the issue too. If the bigger items are filming, a larger green background needs to be used. For instance, a larger green cloth can be used.

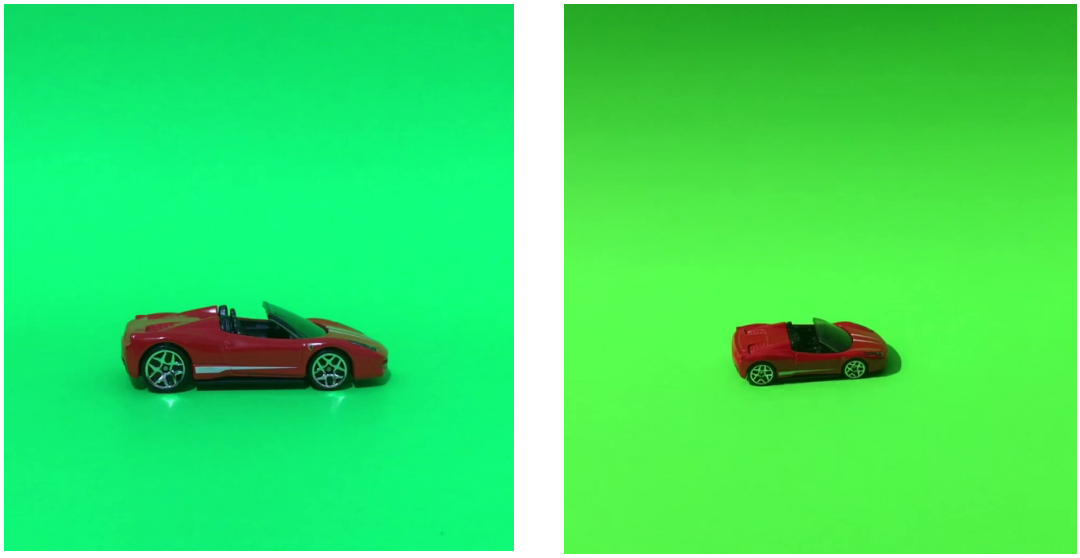


Figure 31: Indoor light and outdoor light

Figure 32: Models in an AR view with different light sources

6.5 Implementation of Tracking the Object in Green Background Videos

After showing customised videos with the green background in a real environment in Section 6.4, we start to think about how to only track and show the object wanted in videos and make the background green. This will enable users to not only use customised videos but also the normal videos like movie clips. This can be useful when thinking of R4 (Blending Videos into reality) in Section 3.1.4 and Step 1D in Section 2.2.1. We are trying to replace most of the background of the videos with counterparts in reality. A library named OpenCV[23] is used to develop the feature. It is used to process videos for background subtracting in Section 6.4. In this section, a workflow of how to track objects in videos and related results are demonstrated. As for testing results, a self-made video (as Figure 13 shows) is used as an example of how we processed a non-customised video. Then we show the result that testing in Hamilton Gardens to show the effect of how the video works in an AR view.

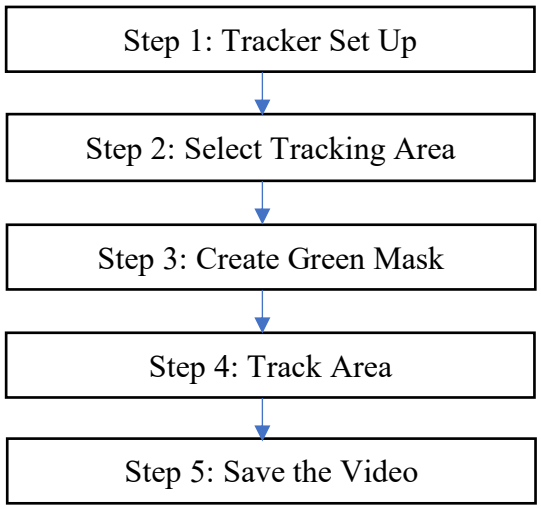


Figure 33: Workflow of tracking the item in a green background video

Five steps of the workflow in Figure 33 are considered when deploying this feature. Firstly, the tracker feature of OpenCV is used for tracking objects in videos. The tracker feature allows us to track a specific area in videos. Before starting to track, a region that users want to track is required for the tracker set-up in Step 2. As Figure 34 shows, a blue box is drawn on the first frame of the video to illustrate the region selected. This is achieved by ROI (Region of Interest) feature of OpenCV. The ROI feature enables users to select an area by using a mouse. The first frame (it can be any frame in the video) of the video is shown on a window for the convenience of users to see which region they want to track. Here, we selected the character as a ROI. Once the ROI is selected, we create a green mask to cover the background that we consider to be redundant in Step 3 as Figure 35 illustrates. This mask later will be turned into transparent when playing

[23] OpenCV, https://opencv.org/about.html

videos in the AR view. Then the video will only show the ROI part. In following Step 4, the program starts to track the area selected. As we can see in Figure 35, the person in the video has a different posture compared to Figure 34. The ROI is consistently tracked while the video is playing. In the final Step 5, we save the processed video on the disk for later use in our application.

Figure 34: Region of Interest

Figure 35: Green mask for covering background

To better show how the processed video behaves in an AR view, we tested processed videos in a real environment. The Lord of the Rings is filmed in the Hobbiton Movie

Set which requires paying for tickets to entry. Also, due to the copyright, we cannot use the movie clip. Because of these factors, we filmed a video in Hamilton Gardens as it is free to enter. The video (as Figure 36 shows) is filmed in front of the fountain in Hamilton Gardens. It is coloured with green except the ROI region as seen from the bottom part of Figure 36. We play it by our application on the same spot where we filmed the video. The application then can overlap the video with only the part we want to show (ROI) with the real environment. By this way, it offers more real reality mixture experience when using a movie clip as we replace the background with the real objects. The actual result can be seen in Figure 37. The virtual video is almost covering the real counterparts, although the scale of the video does not completely match the reality. However, the limitation is that it is hard to detect the boundary of the object rather than selecting manually by ROI currently. Therefore, there are difficulties to only display the person in Figure 36 because we cannot detect the accurate boundary of the person.

Figure 36: Processed video in Hamilton Gardens

Figure 37: Testing the video in our AR application

6.6 Implementation of Multi-tracking Objects in Green Background Videos

Multi-tracking objects simultaneously could bring more possibilities in terms of developing. Thus, based on the single object tracking in a green background video in Section 6.5, multi-tracking version of our software is explored in this section. The OpenCV library offers the multi-tracking feature that enables programs to track many objects at the same time. We first flow the five steps in Figure 33. However, we create different green masks for different objects and ROI areas that are set manually. During the experiment, the green masks overlapping issue emerged. As Figure 38 shows. The user is

supposed to see Object A through the Mask A in the ROI Area as Line A points. However, the Object A is covered by Mask B which is generated for tacking Object B. Users can see nothing but a green video while the objects are still tracked behind the green masks. This issue is addressed by merging all ROI Areas into one green mask on a frame of the video as Figure 39 shows. Firstly, all ROI positions are pre-known as users manually selected them. These ROIs are marked on the same green mask and made into transparent. Finally, all tracking objects (Object A and Object B) can be shown at the same time. To prove the idea, we use the same movie clip in Figure 34 and track the person and the black wooden door. The tracking operation runs successfully as Figure 40 shows. However, the performance of the video processing is relatively poor as two objects are tracking at the same time. If more objects are tracking, the performance could be an issue. This does not affect the mobile application as the video is pre-processed and built-in the application.

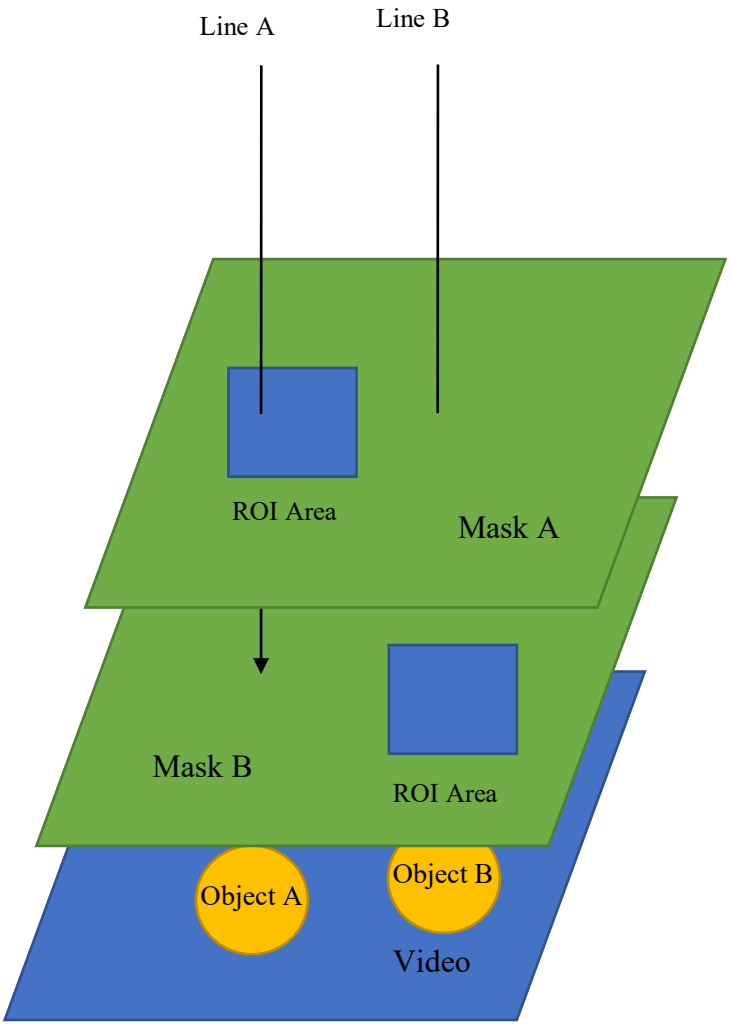


Figure 38: Overlapping green masks.

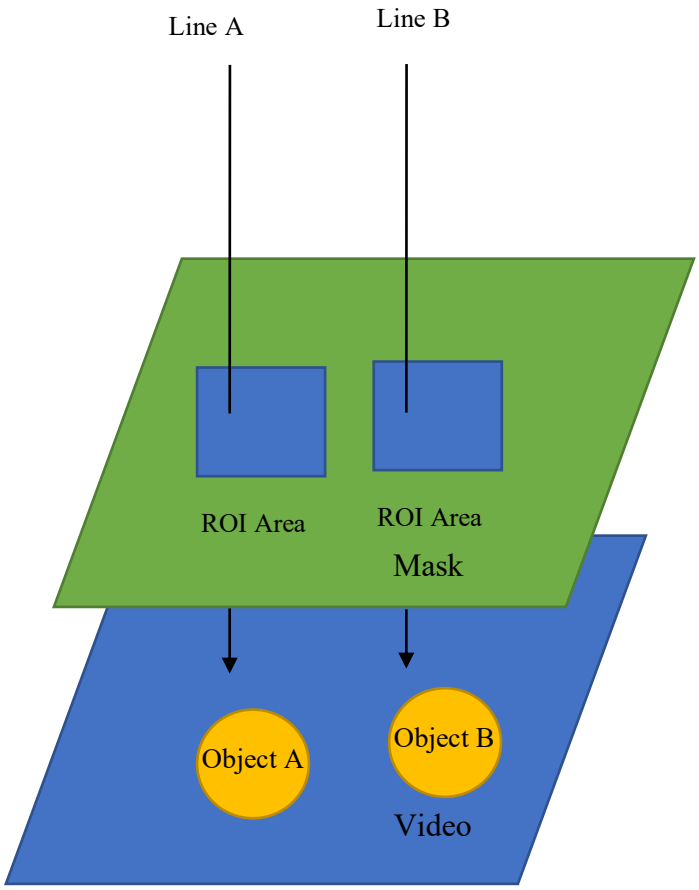


Figure 39: Merging all ROI into one green mask

Figure 40: Multi-tracking in a green background video

6.7 Summary

In this chapter, we explained how our implementation fulfils the core requirements in Section 3.1: Blending Videos into Reality (R4) and Dynamic Ratio (R5), while making use of two types of video sources (customised videos and common videos).

Before the deployment, an analysis of the sample code is conducted, the principle of how to play videos in AR was clearly shown in Section 6.1. Based on this, three modes of R4 were developed. In Section 6.2, Mode I: Video Overlay was deployed by using the Tracker feature of MAXT AR SDK which is based on photo recognition technology. This feature also solves the dynamic ratio of R5. Next, we cut the video into different shapes for better blending videos into reality (Mode II- Video Reshape). Three types of shapes were presented in Section 6.3. They are the rectangular shape which is the default shape for videos to play, solid circle shape for representing shapes with curves and cube shape for three-dimensional shapes. Each of them provides enough credits for proving the video can be drawn into different shapes in AR. After this, Mode III (Video Background Removal) was achieved by the technique of Background Subtraction and the Blending feature of OpenGL in Section 6.4. Background Subtraction helps with separation of the background and the foreground. Blending feature of OpenGL is used to replace the green background of the video with transparent one when playing the video. The limitation was the background separation is not good enough after applying to our application because the part of videos that we want to reserve as the foreground was removed accidentally.

As for video sources, all above experiments were using common videos without modifications. For customised video, we tested a video with a green background filmed by ourselves. This fully meets the need of Mode III much better if the filmed items have less reflective surfaces under the circumstance of the sufficient light.

To extend the Mode III by utilising common videos rather than customised videos, tracking objects selected manually in a green background video was developed. In this case, the wanted part of the video could be reserved while playing in AR and the rest of the video (background) was hidden behind a green mask and turned into transparent. Tracking multi-objects simultaneously in a video and keeping the rest to be transparent are fully supported as well. The result partially supports the Mode III as it cannot remove the background completely. However, the precise boundary of the objects is difficult to track while playing a video. This creates the further requirements for the future.

7 Evaluation

In this chapter, we test our application; analyse the data of tests, then give the results. The evaluation will be conducted from different perspectives of our prototype. Section 7.1 introduces the evaluation perspectives and the methodology. Results are presented in Section 7.2, with a summary in Section 7.3.

7.1 Evaluation Perspectives and Methods

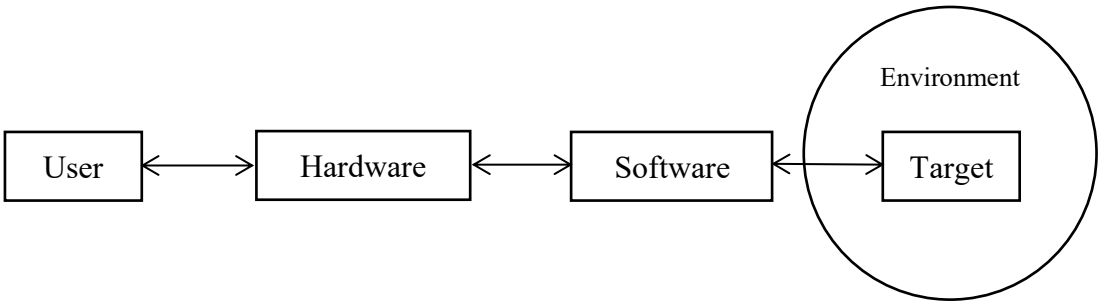


Figure 41: Evaluation perspectives

We will evaluate our prototype from five main perspectives, user, hardware, software, target and environment. The target can be defined as the object to be identified and covered with virtual videos. As the workflow shows in Figure 41, we consider these to be critical parts of the application. Initially, users use hardware (smartphones) for operating software (Our application). The software starts to identify the expected target which could be affected by the environment surrounding. Then the result is delivered back from the target to the user. By analysing each part in Section 7.2, we can gain insights into the quality with which the application performs.

Functional and qualitative testing, and measurements are used for all tests. All tests use real objects for stimulation for actual applications and are conducted without end users but with the researcher carrying out the tests. To get data, the measurement tools such as rulers, protractors and lamps are used across the tests. The result of these tools for measurement is sufficiently accurate for our purposes.

7.2 Result

In this section, we conduct tests according to the evaluation perspectives in Section 7.1. To be more specific, we divided each perspective into several small tasks for ease of testing. Then, the data of tests are analysed; the results are given.

7.2.1 Users

Users can be important for the application. The knowledge of users about how to use smartphones, Augmented Reality should be the prerequisite for using the application. Otherwise, users can be confused before even starting. Mobility is required as people need to move around and scan targets. Another uncertain factor for the application is

holding gesture of mobiles. Different users have different habits for holding mobile phones. In some cases, these gestures can make the application stop working. For example, covering the back camera when holding. In this research, no extra participants are involved in the evaluation. The test is conducted by people with the capability of free moving and rich knowledge about smartphones.

7.2.2 Hardware

Hardware can affect the application. Different devices should be tested in the research. However, as only a few devices are available currently in our research, the test is limited to these devices which will be listed in Table 5, Section 7.2.2.2.

7.2.2.1 Camera Resolution

Theoretically, the more quality camera should make the application working better. This is because AR technology makes use of the stream from the camera. Then images of targets are compared with the frames of the video. The more resolution that frame has, the more likely that the application can recognise the target. More resolution means more detail of the images. However, a video can have various resolutions such as 4K, 1080p, 720p, 480p. Not all cameras are capable to support these resolutions. Additionally, if the SDK is only compatible with the 1080p video which means that only 1080p video can be produced, even via a 4K camera. The resolution that MAXST AR SDK supports is unknown according to their development documents. By reading their latest news[24], they recently start to support up to 1080p resolution and this matches the result of our test result.

Our test is conducted via a Google Pixel with a 12.3 MP back camera. The camera does support 4K video recording. We build a feature which can control the resolution of the video streamed from the camera. A cardboard box (29cm height, 32cm width) is used as a target. The distance between the box and the mobile is measured. As Table 4 shows, the distance of recognition of the target increased along with the resolutions. It stops increasing at 271cm and 1080p, although we have changed the resolution of the camera to 4k. Hence, we conclude that the distance of identification of AR targets benefits from the resolution of the camera. However, it may be limited by the software. Also, the MAXST AR SDK only can make use of 1080p or the lower resolution of the camera. This also can be seen on their website news[24]. They claim they now can support Full HD (1920 x 1080) Video input.

[24] MAXST AR SDK 3.5 Released, https://developer.maxst.com/MD/news/20180319

Video Resolution	Distance of Recognition
480p (640 x 480)	114 cm
720p (1280 x 720)	152 cm
1080p (1920 x 1080)	271 cm
2160p (3840 x 2160)	271 cm

Table 4: Resolution test of cameras

7.2.2.2 Supportive Devices

MAXST AR SDK requires Android OS 4.3 or later to run. There are no requirements for hardware specifically[25]. Theoretically, any Android smart phone with basic specs and an Android OS 4.3 or later can run the SDK without issues. In this research, we tested our application on various devices with different versions of Android OS. As we can see in Table 5, all devices running Android OS 4.3 or later are compatible with our application. However, the latest MAXST AR SDK 3.5[26] is required for running on the recent Android OS. The more recent device can run the application more smoothly, for example, Google Pixel provide more quick response than the Nexus 4 according to our observation. For the speed of the identification of the targets, they all can identify quickly. We have not found the big gap between different versions of the SDK or devices. To make the application working on Android 8.1.0, we upgraded our MAXST AR SDK from 3.0.3 to 3.5.0. The new version of SDK has a few changes which took efforts to upgrade.

Devices	Android OS Version	Support for our application
Google Pixel (2017)	8.1.0	Yes
Nexus 5 (2013)	6.0.1	Yes
Nexus 4 (2012)	4.3	Yes

Table 5: Test on different devices

7.2.3 Software

As we aim to layover videos via AR technology. The types of videos and their sources are important for the actual use in industry or real life. Also, the toolkit for the AR in our application plays an important role. In this section, both video sources and the toolkit are discussed.

[25] MAXST AR SDK requirements and supports, https://developer.maxst.com/MD/doc/3_5_x/supported
[26] MAXST AR SDK 3.5 Released, https://developer.maxst.com/MD/news/20180319

7.2.3.1 Video Sources

We managed to use both of common videos and customised videos as sources for overlaying in an AR view. Here, we recap the issues and key things when using these videos.

A common video is the video that is already filmed by others. It is easy to retrieve them from online or other sources as it is everywhere nowadays. We can lay them on the objects via AR (Mode I: Video overlay, see Section 6.2), even cut them into different shapes, 2D or 3D shapes (Mode II: Video reshape, see Section 6.3) or just reserve the part of the characters in the video like described in Scenario 2 in Section 2.2.2 and in Mode III (Video background removal) in Section 3.1.4. The background subtraction technology from OpenGL does not work properly for our goals. It removes parts that we think are not the background (Section 6.4). The issue then comes to if the user thinks the part is background or not. It is a decision made by users. Different users may have different thoughts and needs for this. We managed to let user select which region they want to reserve when playing videos in AR. The rest of the video is replaced with the transparent background. The region selected is tracking constantly through the video playing (Section 6.5), even for multi-regions (Section 6.6). However, this selected region is in a rectangular shape. This partially meets the needs of Mode III as there is still part of the background left in the region selected. Detection of the exact boundary of objects that user selected is our next goal, but not fulfilled in this research. To completely meet the requirement Mode III (Video background removal) of R4 in Section 3.1.4, our focus moves to another type of video, customised videos.

Customised videos are the video that we filmed by ourselves, then we can decide what we filmed and edit videos for different purposes. As our key focus is only keeping the wanted part in the videos precisely, we first use a demo video which is a person walking with the green background, this works properly as we can see from Section 6.4. This test proves that customised videos fully fulfil the Mode III (Video background removal) requirement in Section 3.1.4 and make Scenario 2 in Section 2.2.2 achievable. After the test, we demonstrated how to film a green background video and found that an object with the less reflective surface is more suitable for filming. The reason is that the reflection area creates a spot with high illumination. The spot can be removed by MAXST AR SDK accidentally with the green the background together. Another factor that can disturb the filming is the light source. This is because the strong direct light can create reflection as well. We recommend using less reflective objects as filming targets and indirect light as light sources when filming.

7.2.3.2 SDK

MAXST AR SDK is the fundamental of our application. We build it into our application. A few issues are found. Spinning issue is found when viewing the cube shape in 6.3.3. Shaking and vanishing issues are identified when the user is far away from the target in Section 6.2 and 6.3. However, the reason can be that a user is viewing the target from an extreme angle or a user is standing too far from the target. These issues are identical with the issues we found in Section 7.2.4.1 and 7.2.4.2. From the test, keeping the size of the target at least two percent of the AR View (the camera view) can reduce the risk

of disappearing targets. Staying in the angle range from 40 degrees to 140 degrees in front of the target can help with the shaking and spinning issues. Nevertheless, MAXST AR SDK is not open source, we cannot possibly know the exact reasons for these issues and fix them without the source code.

7.2.4 Target

Target is the main factor which affects the application significantly. In this section, the main properties of a target that can affect are listed and discussed for the evaluation purpose.

7.2.4.1 Distance

Object	Cardboard box	Matchbox
Height	29cm	3.8cm
Width	32cm	5.4cm
Area	928cm^2	20.52cm^2
Maximum holding distance	281cm	41cm
Minimum holding distance	29cm	7cm

Table 6: Distance of AR targets

Distance is the length between the user and the target. In Section 6.2, the video sticking on targets may shake or even disappear if the user moves away too far or too close. To investigate the reason, we measure the distance between different sizes of objects and the user. A ruler is used for the measurement. We choose a small size object (a matchbox) with 3.8cm height, 5.4cm width and its area is 20.52cm^2 as Table 6 shows. The video can be held steadily when users are away from the matchbox between 7cm to 41cm. Compared to this, the range is much larger for the cardboard box (29cm height and 32cm width), from 29cm to 281cm. Based on these data, the object can be recognized from a longer distance if the object is larger.

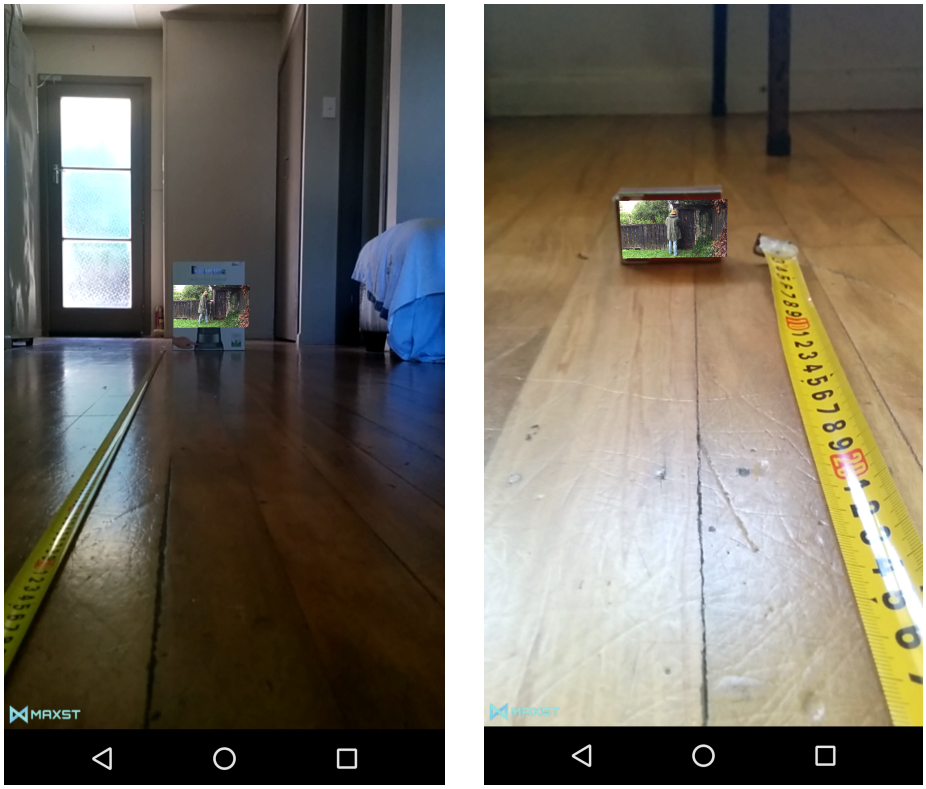


Figure 42: Size of videos in an AR view

After analysis, we think that the success of recognition of objects is more relevant to the size of objects in the AR view than their actual sizes. In Figure 42, we can see that objects with different physical dimensions can be the visually similar size in the AR view. We measure the sizes of them in the AR view in pixels as Table 7 shows. We can see that objects have slightly different sizes but remain the similar ratio of videos' size to the whole AR view. Additionally, the recognition of objects highly relies on the images comparison technology. MAXST AR SDK also analyses the images that we set up as AR targets (see Section 6.2) and compares them with the stream from the camera. We consider the distance that our application can identify objects highly depends on the ratio of objects to the AR view rather than objects' physical sizes. Therefore, if keeping the object in the right ratio to the AR view, the application can identify the targets without problems.

Object	Cardboard box (pixel)	Matchbox (pixel)
Object size	180x220	286x196
Video size	186x104	266x138
AR view Size	1080x1776	1080x1776
Object size/AR view Size	2%	2.9%

Table 7: Measurement of objects in an AR View

7.2.4.2 Angle

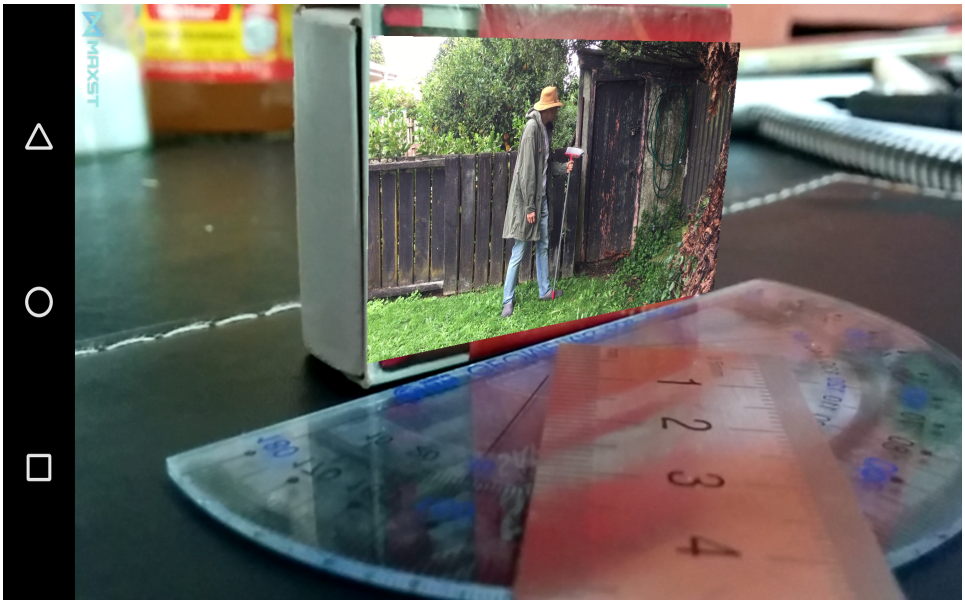


Figure 43: Angle range of recognition targets

Extreme angle is a challenge to AR for recognising and holding the video over targets. As we found our application fails to recognize and hold targets on 180 degrees and zero degree in Section 6.2, we want to test the extreme angles that our application can work. A matchbox is used as a target and mobile phone is placed 10 centre meters away from the target. A protractor and a ruler are used as tools for measurement. As seen in Figure 43, targets can be recognized from 40 degrees to 140 degrees without issues. Once the target is identified, we can move the mobile phone to more wide angles from 10 degrees to 170 degrees. The video is still staying over the target but shaken and spinning occasionally, the closer to the 10 or 170 degrees, the more often the frequency of shaking or spinning is (as Figure 44 presents). However, within the range from 40 degrees to 140 degrees, the application works well. By shooting multi pictures of objects from different angles, the issue is significantly improved like what shows in Figure 45.

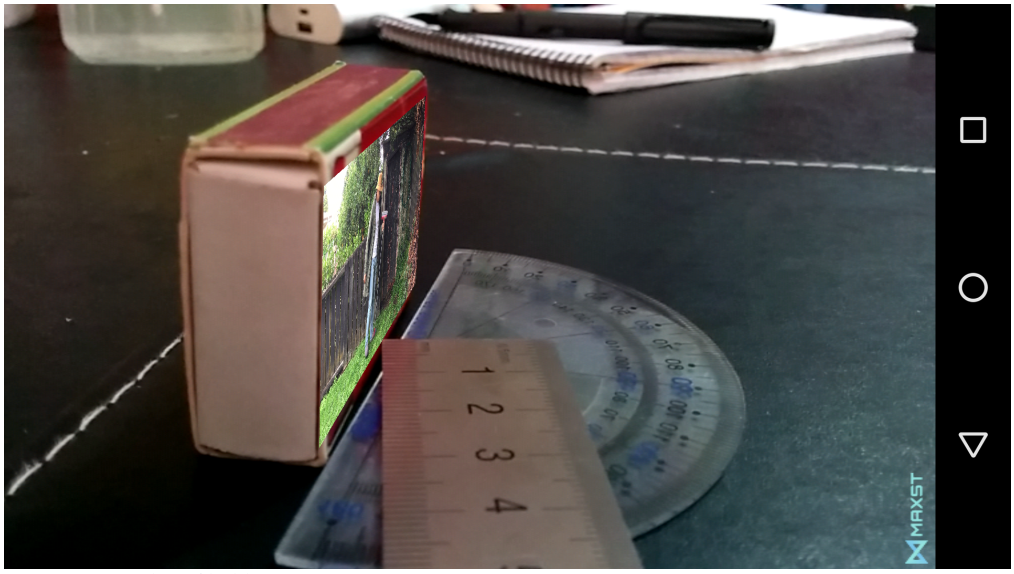


Figure 44: Angles of holding targets in an AR view

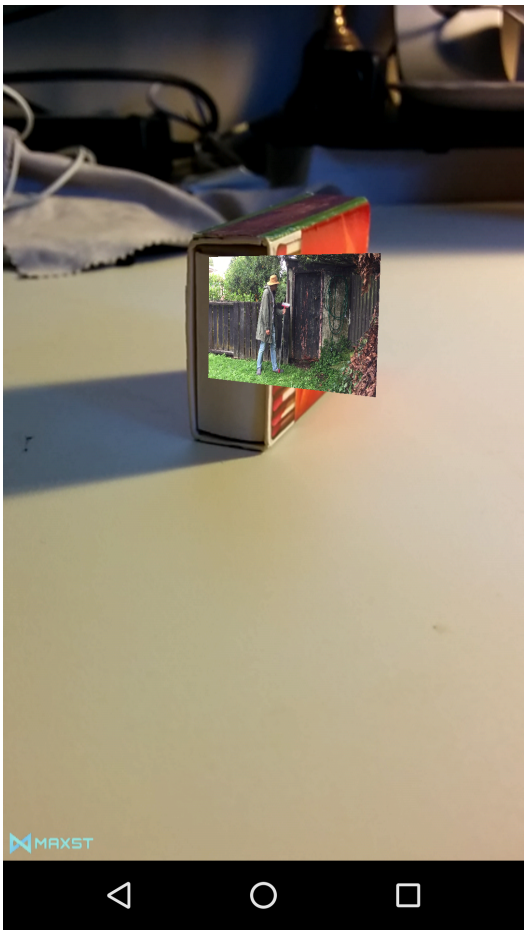


Figure 45: Identification from a side angle

7.2.4.3 Light

Light is another significant part for AR identification. The weak light creates difficulties for the camera to capture the targets and affects the identification. We use the same matchbox box in Table 6 as a test target. The distances from the targets to the user on different light levels are measured. The application Lux Light Meter Pro[27] from App Store is used for measuring light levels. The test result can be seen by following Table 8.

Distance(cm)	Illumination levels (Lux)
25	0
41	3
46	13
47	64

Table 8: Data of light factor test

As seen in Table 8, while the light level is increasing, the maximum distance for identification increases. However, the application, Lux Light Meter Pro shows zero lux when the light is relatively dark, but the camera still can capture the target as Figure 46 shows. With the light level increasing, the recognition distance expands. However, it remains on around 47cm, even with the stronger light.

Figure 46: Testing targets in a dark environment

[27] Lux Light Meter Pro, https://itunes.apple.com/us/app/lux-light-meter-pro/id1292598866?mt=8

We believe sufficient light will increase the identification distance of AR targets. We also noticed a limitation on the distance once the target has enough light. Moreover, in case of uneven strong light, there are possible reflections on the targets, which also introduces further challenges to recognize the target, as we tested in Section 6.4.

7.2.4.4 Cover

Whether the target can still be identified under the circumstance that it is partly covered is now our focus. The same cardboard and matchbox in Table 6 are used. The cardboard is covered about 50 percent and the matchbox with 70 percent respectively. Both can be identified successfully. Any more surface covered fails the application to recognize targets.

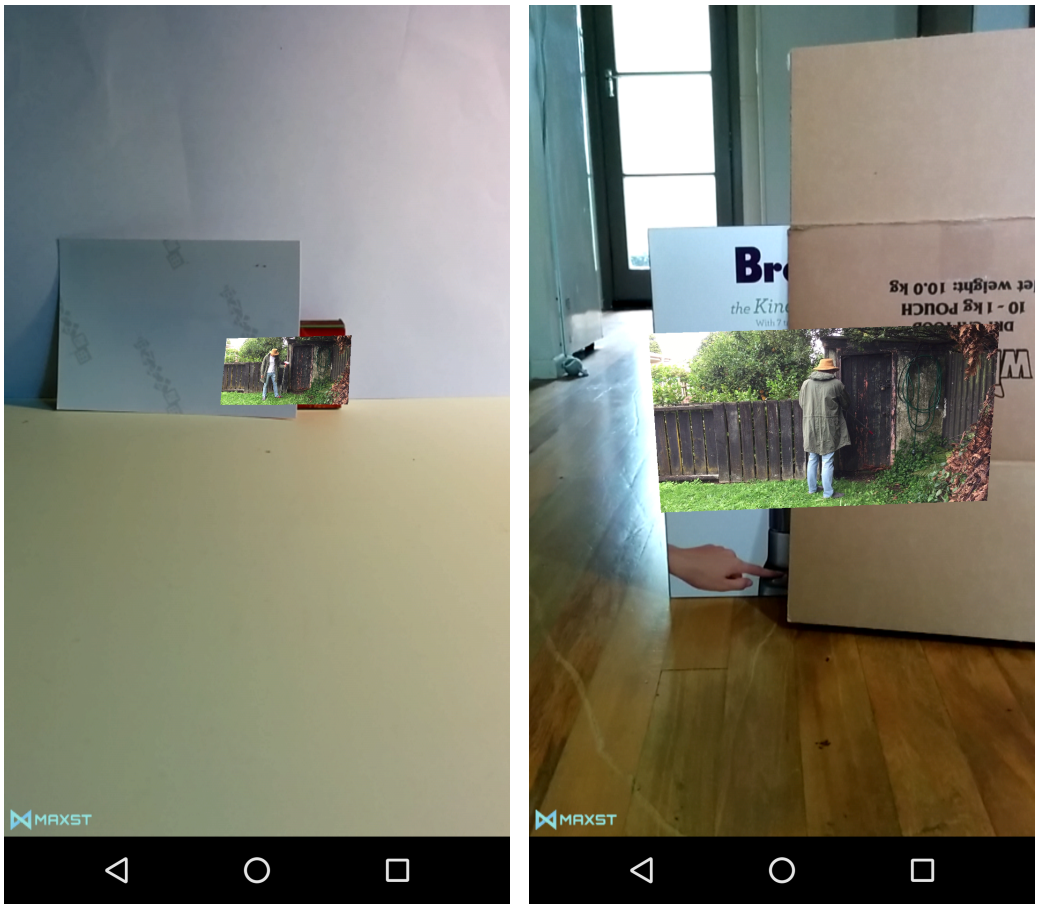


Figure 47: Cover on targets

Another test was deployed. A picture is used as a target with letters “ABCD” and white background. The picture is rated as one star out of five by Target Manager[28] of MAXST, which is a low rate. This means the target is hard to identify by the SDK. However, the application still can put video over the target even the letter D is covered as Figure 48 shows. Any more covered area will fail the application to identify the target. We also tested the repeated letters like “OOOO”. The application could not identify the target

[28] Target Manager, https://developer.maxst.com/MD/doc/g/targetm

this time. This also leads to the next test, texture, because the MAXST claims that the repetitive pattern texture makes difficulties for identification.

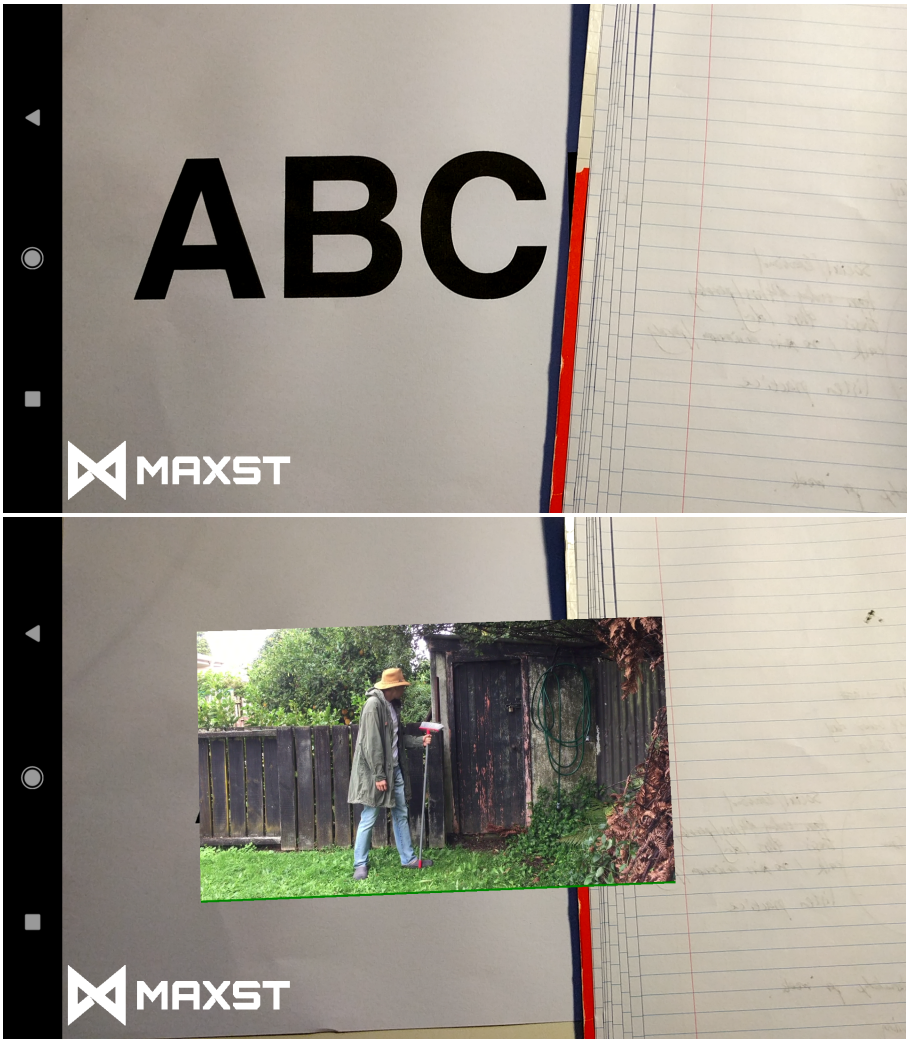


Figure 48: Test cover area on letters

According to our tests, we assume that the identification is related to the feature points rather than the covered area on objects. Once MAXST AR SDK can get enough feature points for recognition, the objects should be identified. However, the threshold of quality and quantity of feature points is unclear in MAXST documents.

7.2.4.5 Texture

As we tested, the texture of a target brings variation when identifying. The target with a repetitive pattern surface is harder to recognize than the one without the repetitive pattern. In the documents of MAXST[29], it points out MAXST Target Manager uses feature

[29] Target Manager, https://developer.maxst.com/MD/doc/g/targetm

points to rate an image target. Repetitive pattern surface often has repetitive feature points. This causes confusion for the image engine to process images.

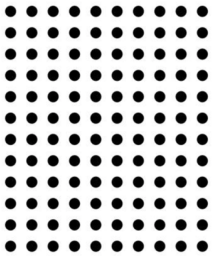
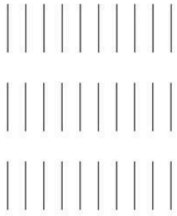
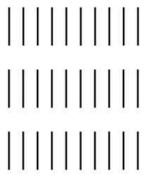
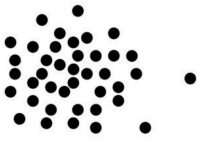
We design several texture patterns for using as targets. In Table 9, we can see images of the pattern and their names. The rate column signifies that how they are rated by MAXST Target Manager. The highest rating is five stars and the lowest is one star. The error means that images cannot be processed by MAXST image learning engine. These images then cannot be built into SDK as AR targets. The MAXST Target Manager suggests replacing[30] the images in this case.

In Table 9, all dots and lines patterns in grid layouts which are repetitive pattern fails to be rated by MAXST Target Manager. All dots and lines in random layouts which are not the repetitive pattern can be identified as AR targets. We first use Grid Dots as the target, then the More Grid Dots. Both are rated as errors. This proves that the number of the unit in patterns is not essential for rating, because the More Grid Dots image has more dots than the Grid Dots image. We then change the shape and the size of the units of the pattern to be lines and bold lines. Both Grid Lines and Grid Bold Lines are failed too. This means the shape of the unit in patterns is not important for the Target Manager rating, so does the size of the shape of the unit if the pattern is repetitive. After this, we tested the Random Dots, Random Lines, Random Bold Lines and Random Dots and Lines. All random shapes patterns can be identified as AR targets successfully. However, the Radom lines are failed to be rated. It is rated as an error image because the lines are too thin; the learning engine cannot get enough feature points. We then use the lines in the same layout and in a bold font (Random Bold Lines). The Random Bold Lines image passes the Target Manager rating test and gets 4 stars.

However, the Random Lines pattern is an exception in the test. We only build the Random Bold Lines into our application database and then scan the Random Lines as an AR target. The target (Random Lines) still can be identified even it does not exist in our database. Also, the Random Lines is rated as an error, so it cannot be built into SDK database. This case gives a hint that Target Manager and SDK may use different methods to visualize the similar AR targets.

In summary, the images with repetitive patterns cannot be rated by Target Manager regardless of the size and number of the unit. They cannot be used by SDK. The images with none respective pattern are suitable to be AR targets. The size of the unit in the pattern does matter in some cases. Some exceptions can emerge. For instance, Target Manager recognises the images as an error, but the images still may be identified by SDK.

[30] Target Manager, https://developer.maxst.com/MD/doc/targetm

Images	Target Pattern	Rate	Identification of Target
	Grid Dots	Error	No
	More Grid Dots	Error	No
	Grid Lines	Error	No
	Grid Bold Lines	Error	No
	Random Dots	3 Stars	Yes

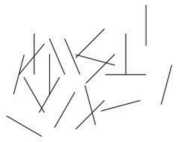
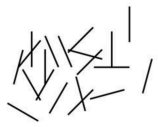
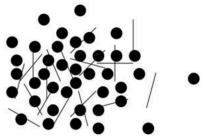
	Random Lines	Error	Yes
	Random Bold Lines	4 Stars	Yes
	Random Dots and Lines	3 Stars	Yes

Table 9: Test texture of targets

7.2.4.6 Colour

MAXST AR SDK claims that they are using feature points for recognition and tracking images[31]. They did not mention the colour of the images. As we tested, the colour is not the main factor for identification of targets. Two same pictures with the colour and without the colour are uploaded to the Target Manager[32] of MAXST. The coloured one is rated with five stars and the black and white one is rated as 4 stars, as seen in Figure 49. Both can be used for identifying targets. We set coloured one as a target in Target Manager and built it into our application. The application should only recognize the colourful target as we set if the colour matters. As we tested, a black and white image of the target was given, and the application immediately identified it. Similarly, a colourful target could be recognized by the application built with the black and white image of the same target. Hence, the colour of the target creates a minor difference in recognizing targets.

[31] What makes a better target image, https://developer.maxst.com/MD/doc/g/targetm
[32] Target Manager, https://developer.maxst.com/MD/doc/g/targetm

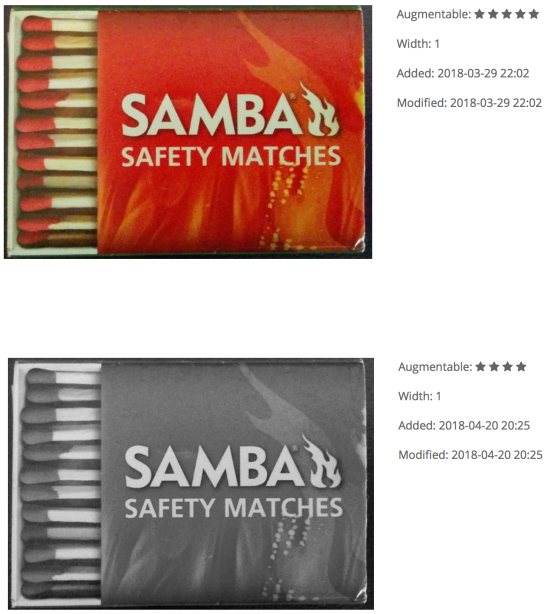


Figure 49: Target colours test

7.2.4.7 Shape

The shape of the target brings no differences when identifying targets as we tested various objects. The MAXST AR SDK is using feature points of the image to compare the target with the stream from the camera. In other words, the application sees a target in pixels rather the overall shape of it.

7.2.5 Environment

The environment can be another crucial factor for the application evaluation. There is a close link between the target and environment. The natural changes can influence targets in terms of the cover area and light on targets. For example, there are many trees in Hamilton Gardens. They may cover the target as they are growing and changing. The cloudy weather can reduce the light on targets so as to fail the reorganisation of the targets.

However, it is difficult to test our application in terms of this factor as it is complicated and varied. Most changes of environment can reflect on targets. By testing the light and the cover on targets, we can stimulate the situation of the environment because of their similarities. The light and the cover test were conducted in Section 7.2.4.3 and 7.2.4.4. More research should be done in the future to explore the environment. In this research, we focus on the targets mainly.

7.3 Summary

In summary, the application completely fulfils the core requirements, R4 and R5 in Chapter 3. Following factors impact on our application significantly.

Users can be a critical factor regarding the way they use the devices and background knowledge about AR and smartphones. All modern devices that running Android OS 4.3 or later should be able to run our application without issues. The higher camera resolution the device has, the more likely the target is recognized, but it may be limited by the development SDK. For the software, all available video sources can be used. An SDK that is not open source like MAXST AR brings difficulties when investigating the programming issues.

The factors of targets are tested and analysed. They can significantly affect the success of identification of the target. Distance, angle, light, cover and texture have more weight than colour and shape in terms of target identification. We found that the distance of successful identification is directly linked to the ratio of targets area to the AR view size on smartphones instead of the actual sizes of targets. Any angle within the range from 40 degrees to 140 degrees is safe for the initial identification. The angle range can be expanded to 10 degrees to 170 degrees once the target is identified. Light is crucial for the camera to capture the images of targets. Sufficient indirect light increases the possibility and the distance to recognize the target. Excessive lighting can create strong reflection which may decrease the chance of identification. About the cover, the target still can be identified, even under the circumstance that the target is partially covered up to 70 percent. This results from the way that MAXST AR SDK compares images with the stream from the camera of the device. Once there are adequate feature points from the target image to compare, the SDK can lay videos over the target steadily. Moreover, the texture with a repetitive pattern on the target reduces the chance to identify the target compared to the one with a none repetitive pattern. However, the colour and shape are secondary in comparison to other factors. Computer vision technology uses feature points as units. The colour and shape bring less influence on the feature points level.

Overall, our application can recognize targets and blend the videos into reality on mobile devices. Various factors should be taken into consideration for recognizing targets. More devices should be tested. More accurate measurement data should be collected in the future. An open source SDK should be explored for an investigation issue we encountered.

8 Discussion

In this chapter, we examine how well our research addresses the issues identified in the introductory chapter. Then our findings are compared with the related work to see how we bring the investigation into a new level. Then particular usages of our application are listed before the summary.

The results of this project show that blending videos into reality is feasible and working well. Our application can lay videos over the real objects via AR technology. Furthermore, it can shape the video and embed a video with a transparent background into reality. This enables users to see a rather real scene. The video with a transparent background can be made either programmatically or manually. The technology of subtracting background is not mature enough to remove the expected background precisely. However, the manual way of filming videos offers rather real user experience. However, the filming can be time-consuming and costly.

8.1 Comparison with Related Work

Related work is reviewed in Chapter 4. We determine to fulfil the requirements, Blending Videos into Reality (R4) and Dynamic Ratio (R5) in Chapter 3, because these related works rarely fully support these features. Our system will be compared with related work reviewed in these requirements for revealing that how our finds refine and extend the video playing in AR. Table 10 is an extension of Table 2 to compare our system, VideoAR, with the systems we reviewed in Chapter 4.

In Section 3.1.4, Blending Videos into Reality (R4) has three modes,

- Mode I: Video overlay.
- Mode II: Video reshape.
- Mode II: Video background removal.

Only Film-location-AR (Park & Woo, 2015) can play videos over the real objects, which is the Mode I, video overlay (as Table 10 shows). However, our system, VideoAR not only supports all three modes but also extends it to the next level, tracking objects in videos. Mode II, video reshape, is supported by our VideoAR prototype, in transforming the video in an AR view into various shapes such as square, solid circle and even a 3D cube. VideoAR also is capable of removing the background during the playing, then deliver a rather real user experience when merging the visual videos and reality (Mode III, video background removal). Single or multi-objects tracking in videos are developed for better removing the background for the extension of Mode III.

Dynamic Ratio (R5) in Section 3.1.5 is supported by The Augmented Video Wall (Baldauf & Fröhlich, 2013). The video can keep the right ratio to the object that the video is attached to. However, other related work fails in this feature. VideoAR fully supports this feature, but it has limitations in terms of the distance and angles. Similar with other systems about the requirement Handheld Device (R9), VideoAR can run on Android devices, which meets the requirement.

		Requirements													
		AR and Video							Geolocation		Mobile	Information management				
		Navigate in AR View	Identify the POIARs	Label the POIARs	Blending Videos into Reality			Dynamic Ratio	Radar	Map	Handheld Device	Search and Browse POIs	Share to Google Map	Share to Social Media	Capture Photos	Record Videos
Categories	**Systems**	R1	R2	R3	R4			R5	R6	R7	R8	R9	R10	R11	R12	R13
					Mode I	Mode II	Mode III									
Superimposing Video on an AR View	**Camera Operator AR**	✔	✔	✖	✖	✖	✖	✖	✖	✖	?	✖	✖	✖	✖	✖
	The Augmented Video Wall	✖	(✔)	✖	✖	✖	✖	(✔)	✖	✖	✔	✖	✖	✖	✖	✖
	Film-location-AR	✔	✔	✔	(✔)	✖	✖	✖	✖	✔	✔	✔	✖	✖	✔	✔
Geolocation and mobile related	**The Touring Machine**	✔	✔	✔	✖	✖	✖	✖	✖	✖	✖	✔	✖	✖	✖	✖
	MobiAR	✔	✔	✔	✖	✖	✖	✖	✖	✖	✔	✔	✖	✖	✔	✖
	AntarcticAR	✔	✔	✔	✖	✖	✖	✖	✔	✔	✔	✖	✖	✖	✖	✖
	VideoAR	✖	✖	✖	✔	✔	✔	✔	✖	✖	✔	✖	✖	✖	✖	✖

Table 10: Comparison of related work. ✔means fully support; (✔) means partially support; ✖ Means not support; ? means unknown support or not.

Other features listed in Chapter 3 were not developed at this stage because our aim was to explore the technical issues and test new ways to blend videos into reality.

8.2 Practical Usages

In this section, we start to explore the practical usages from a technology perspective. Since we have developed the application and fulfilled the goal of this thesis, blending videos into reality. The potential usages in real life of our program are outlined. All these usages are based on our assumption and have not been developed.

8.2.1 Information Presenting

Our application offers the ability to blend rich information such as videos into reality. This brings a new way to present information, especially for the academic field. For example, a student who studies mechanical engineering is assembling a car which is his project. It would be significantly helpful if there is a video tutorial on the scene about how to assemble those complex parts, while using his phone to scan the automobile parts. They can learn by watching videos while doing. Also, they can save time for seeking those tutorials in books because the application helps them to locate the right knowledge according to the real object they want to use the knowledge on. Similarly, Agriculture students can identify plants. A feature as above was developed in Section 6.2. Nonetheless, Video AR brings a more intuitive way to present knowledge.

8.2.2 Information Linking

When starting to think about how to link information with real-life objects, our application, then, would be a considerable candidate. For instance, in the tourist industry, it is often to see a sign to show the history or legends of some buildings, stories or objects like old furniture. Tourists may miss the sign or have to read those long texts. Even

worse, those signs are written in the language that tourists are not familiar with. They may be confused. Compared to this traditional way to link information, our application can present multimedia information over the real object (as seen from Scenario 1 in Section 2.2.1). The video even can be shaped to fit the object (as shown in Section 6.2). Tourists can instantly realize the link between the object in front of them via the videos popped up on their mobiles when scanning. Moreover, the barrier of language is removed to a certain degree because people can get the basic idea by just watching rather than by reading. Meanwhile, they also save time for reading while having fun with watching a short video, as videos offer more comprehensible and rich content than text-based media. Hence, our application can provide an easy way to link information with the real counterpart. The feature is shown in Section 6.2 and 6.3.

8.2.3 Productivity

Testing scenes can be an ideal application for our software. In the film industry, selecting the appropriate scenes can be time-consuming and costly. Some heavy equipment like cameras, crews and even characters need to be sent to the scene and filmed there. Otherwise, a video needs to be edited for testing if the scene is proper. This requires a huge amount of workload. However, these match the scenarios in Chapter 2. By using our application, a phone, a short video of characters or props and a few people are all needed for the test of scenes. Ideally, the video should be with the green background and this would only blend the parts wanted into the scene for the best test result (as we developed in Section 6.4). To save the time for making a green video, any video with parts wanted can be used as well, since we have delivered the feature (single and multi-objects tracking in a green background video) in Section 6.5 and Section 6.6. Similarly, the construction industry also can benefit from this technique. They may need to test their planned buildings on the scene to see how it will be. It is impossible to just put a skyscraper-sized paperboard prototype there. Many of prototypes are done on the computer. However, with our VideoAR technique, a skyscraper is possible to stand just on the scene and give the result intuitively. This also enables builders to see their final work anytime by a lightweight smartphone, which often only can be seen in the office on a PC. Ultimately, our software creates an easy way of testing scenes so as to improve the efficiency of productivity.

8.3 Unexpected Findings

After the evaluation of our application, we find that even the hardware has a 4K resolution camera, the identification of targets is not improved compared to the lower resolution camera, because the SDK of development may restrict the output of the camera. Another valuable finding is the distance of target recognition is highly possible related to the size of the target in an AR view and not its physical size. As AR technology uses feature points to see the object and compare the target image with the stream from the camera, the target image size matters more than its actual size. Based on the same principle, colour and shape make less influence on identification. The repetitive pattern

of texture confuses the MAXST image processing engine. The reason is unknown as the MAXST software is not open source.

8.4 Summary

A full review of our work is discussed throughout this chapter. A comparison with other similar work shows what gaps we filled. Then, we assess the applications of our research in real life in the fields such as academia, tourism, film and architecture. We also present our unexpected findings for the further research.

9 Conclusion

This chapter presents a summary of the achievements of this thesis and the limitations of the study, and provides a brief discussion of future work.

9.1 Summary

The goal and the context of this research were examined in Chapter 1. The goal was to explore the feasibility of blending video-based content into reality via AR technology and mobile phones. Chapter 2 set a persona and two scenarios for identifying the requirements. Based on the persona and scenarios, thirteen features were abstracted in Chapter 3. Several related systems were reviewed and compared in Chapter 4 according to the requirements in Chapter 2. In Chapter 5, we started to design the system and explored the available mobile operating systems, frameworks and libraries before we developed the prototype. In Chapter 6, the implemented prototype was described based on the design work in Chapter 5. The prototype was evaluated, and Chapter 7 presents the results of the evaluation of the performance and discusses the limitations we identified. Then in Chapter 8, we discussed the practical usages, unexpected findings and what we achieved compared to other systems reviewed in Chapter 4. We achieved three complex features of thirteen features designed in Chapter 3 that other systems failed to fulfil.

To conclude this thesis, we explored the feasibility of combining technologies of Augmented Reality, mobile devices and videos. We presented a new way to produce, present and link the information with the real objects. We developed a prototype that shows the video in AR can be manipulated in various ways. However, on the way towards the combination of these technologies, a number of limitations were identified. Despite its limitations, the combination of video, mobile development and AR technology is promising. Our system will show better quality as the AR libraries are developed further and become more reliable, and cameras became more powerful. The system will then be appropriate for industrial applications to serve the needs of tourism, architecture and film fields.

The source code, the demo videos and related pictures are accessible via GitHub (https://github.com/xiaoxin628/VideoAR.git)

9.2 Future Work

Further work is expected in following fields.

Accuracy of background subtractions

In Section 6.4, we attempted to remove the background of the video. However, the result did not completely fulfil our requirement R4, Mode III (Video background removal) in Section 3.1.4. To better automatically separate the foreground from the background, research into improved accuracy of background subtraction is needed.

Efficiency of filming green background videos

Currently, filming a green background video is time-consuming and limited by many factors such as reflection of surface, light and size of objects (as seen in Section 6.4). A more efficient way to film a video with a green background that does not need experts or commercial setups should be explored in the future, especially for larger objects.

Open source framework

We experienced a number of issues that could not be explored or addressed when developing the prototype because the library is sealed and acts as a black box. For example, we could not know the details of the algorithm that was used to identify objects and what the extremes of identifying objects are in terms of distance, angle, light, cover, texture, colour and shape (Section 7.2.4). We also had difficulties in customising features and investigating bugs because the source code was not available to us. If possible, an open source library or framework should be considered when developing AR features further. Unfortunately, no open source library of sufficient quality was available during this project.

Undelivered features

In this research, thirteen features were designed in Section 3. R4 (Blending videos in to Reality, 3.1.4) and R5 (Dynamic Ratio, 3.1.5) of these features were developed. This is because we were aiming to investigate the feasibility of playing videos in AR on mobile platforms and these two features are the core elements of the study. Further development of these omitted features would allow user-based testing of the AR features in context.

Devices

Three devices (Nexus 4, 2012, Nexus 5, 2013 and Google Pixel, 2017) were tested in this research as listed in Section 7.2.2.2. More and newer devices could be tested in the future to ensure the applicability for a range of available hardware.

References

Appiah, O. (2006). Rich Media, poor Media: The Impact of audio/video vs. text/picture testimonial ads on browsers' evaluations of commercial web sites and online products. *Journal of Current Issues & Research in Advertising, 28*(1), 73-86.

Arakawa, T., Kasada, K., Narumi, T., Tanikawa, T., & Hirose, M. (2013). *Augmented reality system for overlaying a scene in a video onto real world and reliving the camera operator's experience.* The IEEE Virtual Reality (pp. 139-140). Lake Buena Vista, FL USA: IEEE

Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., & MacIntyre, B. (2001). Recent advances in augmented reality. *IEEE computer graphics and applications, 21*(6), 34-47.

Baldauf, M., & Fröhlich, P. (2013). *The augmented video wall: Multi-user AR interaction with public displays.* CHI'13 Extended Abstracts on Human Factors in Computing Systems (pp 3015-3018). New York, NY, USA: ACM.

Bouwmans, T. (2012). Background subtraction for visual surveillance: A fuzzy approach. In S. K. Pal, A. Petrosino & L. Maddalena (Eds.), *Handbook on soft computing for video surveillance (pp. 103-138)*. Boca Raton, FL: Chapman & Hall/CRC.

Broll, W., Lindt, I., Herbst, I., Ohlenburg, J., Braun, A.-K., & Wetzel, R. (2008). Toward next-gen mobile AR games. *IEEE Computer Graphics and Applications, 28*(4), 40-48.

Carroll, J. M. (2000). Five reasons for scenario-based design. *Interacting with computers, 13*(1), 43-60.

Cooper, A. (2004). Designing for Pleasure. *The inmates are running the asylum:Why high-tech products drive us crazy and how to restore the sanity*. Retrieved from http://ptgmedia.pearsoncmg.com/images/9780672326141/samplepages/0672326140.pdf

Dünser, A., Billinghurst, M., Wen, J., Lehtinen, V., & Nurminen, A. (2012). Exploring the use of handheld AR for outdoor navigation. *Computers & Graphics*, 36(8), 1084-1095.

Feiner, S., MacIntyre, B., Höllerer, T., & Webster, A. (1997). A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment. Personal Technologies, 1(4), 208-217.

García-Crespo, A., Chamizo, J., Rivera, I., Mencke, M., Colomo-Palacios, R., & Gómez-Berbís, J. M. (2009). SPETA: Social pervasive e-Tourism advisor. *Telematics and Informatics, 26*(3), 306-315.

Hamilton Gardens New Zealand. (2017). About us. Retrieved from http://hamiltongardens.co.nz/about-us/

International Data Corporation. (2017). Smartphone OS market share, 2017 Q1. Retrieved from https://www.idc.com/promo/smartphone-market-share/os

Jackson, P., Osborne, B., & Walsh, f. (Producer), &Jackson, P. (Director). (2001). The *Lord of the Rings: The Fellowship of the Ring*[Motion Picture]. USA: New Line Cinema.

Kounavis, C. D., Kasimati, A. E., & Zamani, E. D. (2012). Enhancing the tourism experience through mobile augmented reality: Challenges and prospects. *International Journal of Engineering Business Management, 4*, 1-6.

Lee, G. A., Dunser, A., Nassani, A., & Billinghurst, M. (2013). *AntarcticAR: An outdoor AR experience of a virtual tour to Antarctica.* Mixed and Augmented Reality-Arts, Media, and Humanities (ISMAR-AMH), IEEE International Symposium. Retrieved from https://ieeexplore.ieee.org/document/6671264/

Marimon, D., Sarasua, C., Carrasco, P., Álvarez, R., Montesa, J., Adamek, T., … Gascó, P. (2010). *MobiAR: tourist experiences through mobile augmented reality*. Barcelona, Span: Telefonica Research and Development.

Macionis, N. (2004). Understanding the film-induced tourist. In International tourism and media conference proceedings (Vol. 24, pp. 86-97). Melbourne, Australia : Tourism Research Unit, Monash University.

OpenCV. (2017). Background subtraction. Retrieved from https://docs.opencv.org/3.3.0/db/d5c/tutorial_py_bg_subtraction.html

OpenGL. (2017). Transparency sorting. Retrieved from https://www.khronos.org/opengl/wiki/Transparency_Sorting

Park, H., & Woo, W. (2015). *Metadata design for location-based film experience in augmented places.* In Mixed and Augmented Reality-Media, Art, Social Science, Humanities and Design (ISMAR-MASH'D), IEEE International Symposium (pp. 40-45). Fukuoka, Japan: IEEE.

Royalty Free Stock Footage. (2016, May 4). Man talking on the phone against green screen. Stock footage [Video file]. Retrieved from https://www.youtube.com/watch?v=_7el7zSSdws

Singh, K., & Best, G. (2004). Film-induced tourism: Motivations of visitors to the Hobbiton movie set as featured in the Lord of the Rings. In W. Frost, G. Croy (Eds), *International Tourism and Media Conference Proceedings (pp.* 98-111*)*. Melbourne: Tourism Research Unit, Monash University.

Unity. (2017). The world-leading creation engine for gaming. Retrieved from https://unity3d.com

Vuforia. (2017). Vuforia. Retrieved from https://www.vuforia.com

Wikitude. (2017). See beyond reality! With Wikitude SDK7. Retrieved from https://www.wikitude.com

Yovcheva, Z., Buhalis, D., & Gatzidis, C. (2012). Smartphone augmented reality applications for tourism. *E-review of tourism research, 10*(2), 63-66.

Zivkovic, Z. (2004). Improved adaptive Gaussian mixture model for background subtraction. Proceedings of the 17th International Conference on Pattern Recognitio*n* (pp. 28-31). Cambridge, England: IEEE.

Zivkovic, Z., & Van Der Heijden, F. (2006). Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters, 27*(7), 773-780.